Simulation of a Moving Elastic Beam Using Hamilton's Weak Principle

THESIS

Elliott Johnathon Leigh, First Lieutenant, USAF

AFIT/GAE/ENY/06-M21

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GAE/ENY/06-M21

Simulation of a Moving Elastic Beam Using Hamilton's Weak Principle

THESIS

Presented to the Faculty

Department of Aeronautical and Astronautical Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Elliott Johnathon Leigh, B.S.A.E.

First Lieutenant, USAF

March 2006

AFIT/GAE/ENY/06-M21

Simulation of a Moving Elastic Beam Using Hamilton's Weak Principle

Elliott Johnathon Leigh, B.S.A.E.
First Lieutenant, USAF

Approved:

/signed/	6 Mar 2006
Dr. Donald Kunz (Chairman)	date
/signed/	6 Mar 2006
Dr. Robert Canfield (Member)	date
/signed/	6 Mar 2006
Dr. Anthony Palazotto (Member)	date

AFIT/GAE/ENY/06-M21

Abstract

Hamilton's Law is derived in weak form for slender beams with closed cross sections. The result is discretized with mixed space-time finite elements to yield a system of nonlinear, algebraic equations. An algorithm is proposed for solving these equations using unconstrained optimization techniques, obtaining steady-state and time accurate solutions for problems of structural dynamics. This technique provides accurate solutions for nonlinear static and steady-state problems including the cantilevered elastica and flatwise rotation of beams. Modal analysis of beams and rods is investigated to accurately determine fundamental frequencies of vibration, and the simulation of simple maneuvers is demonstrated.

Acknowledgements

Research is neither an individual effort nor accomplishment; its fruit is the result of collective thinking, trial, and error. This thesis is no exception, and I must credit the authors of three decades of previous work in this field for laying the foundation. My particular thanks go to Dr. Donald Kunz, my advisor, who spent many hours teaching me the complex dynamics required for an undertaking of this magnitude, which cannot be learned from a textbook alone. Also, to LtCol Mark Abramson, whose expertise in the area of nonlinear optimization techniques was crucial to building an algorithm that is not only correct in theory, but robust and practical enough to solve realistic problems.

Elliott Johnathon Leigh

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures ix

List of Tables xii

List of Symbols xiii

List of Abbreviations xv

I. Introduction 1

II. Background 3
- 2.1 History of Multibody Systems Analysis 3
- 2.2 Adding Flexibility to System Components 4
- 2.3 DAE Methods 5
- 2.4 Hamilton's Law 6
- 2.5 Finite Elements in Space and Time 8
- 2.6 Hamilton's Principle 9
- 2.7 Hamilton's Weak Principle 10
- 2.8 Context of the Current Work 12

III. Theory 14
- 3.1 Conventions 14
- 3.2 Assumptions 15
- 3.3 Coordinate Reference Frames 15
- 3.4 Rotation Parameters 16
- 3.5 Beam Geometry 18
- 3.6 Beam Kinematics 20
- 3.7 Virtual Displacements and Rotations 22
- 3.8 Generalized Speeds and Strains 23
- 3.9 Kinetic and Potential Energy 26
- 3.10 Generalized Momenta, Forces, and Moments 28
- 3.11 Virtual Work and Virtual Action 30
- 3.12 Derivation of Equations from Hamilton's Law 31
- 3.13 Finite Element Discretization 36

Page

3.14 Equations for Steady-State Analysis 47
3.15 Solution Algorithm 48
3.16 Scaling and Units 50
3.17 Choosing the Initial Guess 51
3.18 Analytical Jacobian 52
3.19 Error Tolerance Criteria 53
3.20 Conservation Laws 53

IV. Analysis and Results 55
4.1 Axial Rods 55
4.2 Timoshenko Beams 61
4.3 Cantilevered Elastica 67
4.4 Beam with a Follower Force 69
4.5 Beam Rotating at a Constant Angular Velocity 72
4.6 Rotating Beam with Applied Loads 77
4.7 Axial Vibration of Rods 78
4.8 Torsional Vibration of Shafts 84
4.9 Transverse Vibration of Beams 87
4.10 Spin Up Maneuver 88
4.11 Flapping Maneuver 90

V. Conclusions 93
5.1 Summary of Results 93
5.2 Recommendations for Future Research 94

Appendix A. Solutions to Steady-State Problems 97

Appendix B. Solutions to Free Vibration Problems 109

Appendix C. Solutions for Basic Maneuvers 125

Appendix D. Matlab Code 128
D.1 Executable.m 128
D.2 Geomat.m 129
D.3 Simparams.m 130
D.4 FEStatic.m 132
D.5 Static.m 133
D.6 FEIncStatic.m 142
D.7 IStatic.m 144
D.8 FEDynamic.m 152
D.9 Dynamic.m 153
D.10 Milenkovic.m 163

Page

D.11 Tilde.m 163
D.12 ImportStatic.m 164
D.13 ImportDynamic.m 166

Bibliography 169

List of Figures

Figure		Page
3.1.	Geometry Schematic	19
3.2.	Finite Element Depiction	40
3.3.	Time Marching Procedure	49
3.4.	Initial and Boundary Conditions	50
4.1.	Axial Rod Configurations	56
4.2.	Deformation of an Axial Rod with Tip Loads	57
4.3.	Deformation of an Axial Rod with Distributed Loads	59
4.4.	Convergence of Error in Rod Problems	60
4.5.	Modeling of a Tapered Rod	61
4.6.	Deformation of a Tapered Rod with Tip Loads	61
4.7.	Timoshenko Beam with Transverse Loads	62
4.8.	Convergence of Error in Timoshenko Beam Problems	63
4.9.	Deformation of a Timoshenko Beam with Transverse Shear Loads	64
4.10.	Timoshenko Beam with Bending Moments	65
4.11.	Deformation of a Timoshenko Beam with Bending Moments	66
4.12.	Cantilevered Elastica Solution	68
4.13.	Convergence of Error in the Cantilevered Elastica Problem	69
4.14.	Follower Force Problem	70
4.15.	Follower Force Solutions	71
4.16.	Steady Rotating Beam Problem	72
4.17.	Convergence of Rotating Beam Solutions	76
4.18.	Rotating Beam with a Distributed Load	77
4.19.	Displacement of a Rotating Beam with a Distributed Load.	79
4.20.	Propagation of Force Wave through a Fixed-Free Rod	80
4.21.	Effect of Element Size on Single Element Frequency Response	82

Figure		Page
4.22.	Effect of Element Size on Multiple Element Frequency Response	83
4.23.	Tip Displacement of a Vibrating Axial Rod	83
4.24.	Effect of Step Size on Propagation of Force Wave	84
4.25.	Numerical Conservation of Energy	85
4.26.	Effect of Step Size on Propagation of Torque Wave	86
4.27.	Frequency Response of Torsional Vibration Problem	86
4.28.	Frequency Response of Bending Vibration Problem	88
4.29.	Time History of Spin Up Maneuver	89
4.30.	Frequency Response of Spin Up Maneuver	90
4.31.	Time History of Flapping Maneuver	91
4.32.	Frequency Response of Flapping Maneuver	92
A.1.	Solution for an Axial Rod with Tip Loads	97
A.2.	Solution for an Axial Rod with Distributed Loads	98
A.3.	Solution for a Tapered Rod with Tip Loads	99
A.4.	Solution for a Timoshenko Beam with a Tip Load	100
A.5.	Solution for a Timoshenko Beam with a Distributed Load	101
A.6.	Solution for a Timoshenko Beam with a Distributed Bending Moment	102
A.7.	Solution for a Timoshenko Beam with a Tip Bending Moment	103
A.8.	Solution for the Follower Force Problem	104
A.9.	Solution for a 12 inch, Rotating Aluminum Beam	105
A.10.	Solution for a 12 inch, Rotating Elastic Beam	106
A.11.	Solution for a 72 inch, Rotating Aluminum Beam	107
A.12.	Solution for the Rotating Beam with a Distributed Load	108
B.1.	Axial Vibration of a Fixed-Free Rod, $N_x = 10$, $\Delta t = 0.006ms$	109
B.2.	Axial Vibration of a Fixed-Free Rod, $N_x = 10$, $\Delta t = 0.003ms$	110
B.3.	Axial Vibration of a Fixed-Free Rod, $N_x = 20$, $\Delta t = 0.006ms$	111
B.4.	Axial Vibration of a Fixed-Free Rod, $N_x = 1$, $\Delta t = 0.005ms$	112

Figure Page

B.5. Torsional Vibration of a Fixed-Free Rod,$N_x = 10$, $\Delta t = 0.005ms$ 113

B.6. Torsional Vibration of a Fixed-Free Rod,$N_x = 10$, $\Delta t = 0.01ms$ 114

B.7. Torsional Vibration of a Fixed-Free Rod,$N_x = 1$, $\Delta t = 0.005ms$ 115

B.8. Torsional Vibration of a Fixed-Free Rod,$N_x = 1$, $\Delta t = 0.1ms$. 116

B.9. Torsional Vibration of a Fixed-Free Rod,$N_x = 20$, $\Delta t = 0.01ms$ 117

B.10. Bending Vibration of a Cantilevered Beam,$N_x = 10$, $\Delta t = 1ms$ 118

B.11. Bending Vibration of a Cantilevered Beam,$N_x = 10$, $\Delta t = 0.145ms$ 119

B.12. Bending Vibration of a Cantilevered Beam,$N_x = 10$, $\Delta t = 16ms$ 120

B.13. Bending Vibration of a Cantilevered Beam,$N_x = 20$, $\Delta t = 1ms$ 121

B.14. Bending Vibration of a Cantilevered Beam,$N_x = 20$, $\Delta t = 0.145ms$ 122

B.15. Bending Vibration of a Cantilevered Beam,$N_x = 20$, $\Delta t = 16ms$ 123

B.16. Conservation of Energy for Bending Vibration Problems 124

C.1. Spin Up Maneuver (z axis of rotation),$N_x = 10$, $\Delta t = 0.5ms$ . . 125

C.2. Flapping Maneuver (y axis of rotation),$N_x = 10$, $\Delta t = 0.2ms$, $E = 10e6\,psi$ 126

C.3. Flapping Maneuver (y axis of rotation),$N_x = 10$, $\Delta t = 0.2ms$, $E = 10e3\,psi$ 127

List of Tables

Table		Page
4.1.	Axial Rod Convergence	59
4.2.	Tapered Rod Convergence	60
4.3.	Timoshenko Beam Convergence	64
4.4.	Cantilevered Elastica Convergence	68
4.5.	Follower Force Convergence	71
4.6.	Rotating Beam Convergence	75
4.7.	Loaded, Rotating Beam Convergence	78

List of Symbols

Symbol		Page
L	Lagrangian	6
δW	Virtual Work	6
q_k	Generalized Coordinate	6
C^{Bb}	Direction Cosine Matrix	16
H^{Bb}	Rotation Parameter Matrix	16
H_B	Angular Momentum	16
$\rho_{B/b}$	Weiner-Milenkovic Rotation Parameters	17
r	Undeformed Beam Reference Line	18
x_1	Undeformed Reference Line Coordinate	18
R	Deformed Beam Reference Line	18
s	Deformed Reference Line Coordinate	18
$\bar{r}$	Undeformed Position	18
$\bar{R}$	Deformed Position	18
$\bar{u}$	Deformation	18
ξ_b	Cross Sectional Position	19
$w_i B_i$	Warping	19
V_B	Deformed Velocity	20
v_b	Undeformed Velocity	20
ω_b	Undeformed Angular Velocity	20
Ω_B	Deformed Angular Velocity	20
γ	Force Strain	21
κ	Moment Strain	21
K_B	Deformed Curvature	21
k_b	Undeformed Curvature	21
q	Generalized Displacement	22

Symbol		Page
ψ	Generalized Rotation	22
δq	Virtual Displacement	22
$\delta\psi$	Virtual Rotation	22
T	Kinetic Energy Per Unit Length	26
m	Mass Per Unit Length	26
I	Mass Moment of Inertia	26
$\bar{\xi}_B$	Center of Mass Offset	26
U	Potential Energy Per Unit Length	27
g_B	Gravity Vector	27
P_B	Generalized Momentum	28
H_B	Generalized Angular Momentum	28
F_B	Generalized Force	29
M_B	Generalized Moment	29
E	Modulus of Elasticity	30
G	Shear Modulus	30
J	Torsional Rigidity	30
A	Cross Sectional Area	30
K	Effective Shear Area	30
f_B	Applied Force Per Unit Length	30
m_B	Applied Moment Per Unit Length	30
$\overline{\delta A}$	Virtual Action	31
λ_n	Lagrange Multiplier	31
τ	Nondimensional Time Step	37
ε	Nondimensional Length	37
X_i	State Vector	52

List of Abbreviations

Abbreviation		Page
MSA	Multibody Systems Analysis	1
HWP	Hamilton's Weak Principle	1
DAE	Differential Algebraic Equation	5
FFT	Fast Fourier Transform	80
PDE	Partial Differential Equation	85

Simulation of a Moving Elastic Beam Using Hamilton's Weak Principle

I. Introduction

Multibody systems analysis (MSA) is an analytical tool used to solve problems of dynamics for complex mechanical systems. Common implementations in software represent a system by a series of rigid bodies connected with joints, where differential equations of motion are coupled with algebraic constraints and solved numerically. Rigid body motion is a useful simplifying assumption in mechanics; however, it is not sufficient for modern applications with highly elastic materials undergoing large motions. Some examples include rotor blades, flexible wings on aircraft, elastic linkages, satellites with flexible arms, and flapping wings.

There are many techniques for modeling the behavior of dynamic systems with flexible components. Chapter 2 highlights some of the various numerical methods in the literature and their specific applications. The purpose of this research is to explore the use of Hamilton's Weak Principle (HWP) as a unifying theory for the dynamic simulation of both flexible and rigid body motion. It is intended to serve as a proof of concept by applying the algorithm to the structural dynamics of beams, which extends directly to aircraft wings, rotor blades, robot arms, and structural members (spars, stringers, struts, etc.). The approach offers many advantages: it does not require a differential equation solver, constraints are incorporated directly into the problem with Lagrange multipliers, and a finite element model with simple shape functions may be used.

There are three tasks involved in this research. First, a mixed, space-time, finite element formulation for beams is derived using Hamilton's Weak Principle. While an intrinsic formulation for elastic beams has been developed in the literature [26],

this work discretizes the resulting equation in space and time with finite element techniques. This process transforms a complex integral equation into a system of nonlinear algebraic equations which are solved numerically. The mixed formulation is unique in its inclusion of deformations, momenta, internal resultants, and generalized speeds and strains as independent field variables which are evaluated simultaneously.

Next, an algorithm is developed to assemble and solve the system of nonlinear algebraic equations with existing optimization techniques. The complete configuration of the body is determined at each time step, resulting in a time marching algorithm. Finally, the solver is tested against several problems of mechanics to evaluate its robustness. The second and third tasks are iterative, as improvements to the algorithm are required in order to solve more complex problems. Initially, static and steady-state solutions to problems with various degrees of complexity and nonlinearity are obtained and compared with exact solutions to verify accuracy. Finally, the algorithm is used to obtain time accurate solutions to problems of beam dynamics.

II. Background

2.1 History of Multibody Systems Analysis

The term "multibody system" refers to a complex mechanical system which may be represented by an equivalent model of discrete, interconnected bodies [50]. Examples include automobiles, machinery, engines, robotic devices, and aerospace vehicles. The individual components are represented numerically by their material and geometric properties, and are connected and constrained by various classes of joints.

In the 1960s, the digital computer made numerical analysis possible for complex multibody systems. This led to the development of general purpose computer programs for MSA. However, the early versions were limited to two-dimensional systems with open tree configurations (where a cut in any component separates the system in half) [48]. This limited the application to systems which are relatively simple.

The ability to treat closed circuit systems as well as three-dimensional motion evolved in the 1970s to treat more complex systems such as spacecraft. Research was also geared toward the simulation and design of large scale systems whose components experienced large angular rotations (turbomachinery, camshafts, flywheels, etc.). More complex systems required the simultaneous solution of hundreds to thousands of differential equations. Many computational techniques were developed for this purpose using rigid body or gyrostatic motion as a basic assumption for individual components. While no mechanical system exhibits pure rigid body motion, this assumption can simplify the problem, reducing computational cost without losing much accuracy.

As greater emphasis was placed on "high-speed, lightweight, precision systems," the need to include elasticity in the analysis became more important [50]. Modern machinery operates at high speeds, high temperatures, in hostile environments, and is designed to high tolerances. Neglecting deformation effects for these types of systems will lead to a poor mathematical representation. Additionally, trends in the aerospace industry are toward lightweight structures and elastic materials capable of

large deformations. Enforcing the rigid body assumption would restrict the motion of flexible components and invalidate the simulation. A realistic math model must include the effects of flexibility in the system equations.

2.2 Adding Flexibility to System Components

Equations governing nonlinear, flexible, multibody systems are far more complex than those for rigid body motion. A rigid body can be completely described by a position and orientation vector, which are propagated forward in time based on initial conditions and applied forces and moments. Constraints imposed by joints can be introduced with Lagrange multipliers. All system information is included in the differential equations of motion and the algebraic constraints, which are numerically integrated in time.

With flexible systems, all of the above applies and more. A rigid body can be completely described by the motion of a single particle within the body and its position from the center of gravity, but a flexible body requires complete configuration data for every material point in the body. Because every particle can move relative to one another, integration is required in both space and time to obtain complete information. Further complexities appear in the geometric and material nonlinearities that can arise. Robust algorithms are necessary to perform time integration of these types of systems [11].

Early techniques for incorporating flexible body dynamics involved the use of floating reference frames [16, 50] and finite elements with convected coordinate systems [13, 32]. Moving reference frames are attached to rigid body motion and linear elastic deformation theories are applied to the discrete components of the system. Deformations are described relative to the rigid body motion using these intermediate coordinate systems. Software codes were modified to include both flexible and rigid bodies, but the early methods were not well suited for systems with large deformations and sometimes nonlinear effects were difficult to capture [20].

In the 1980s, the introduction of finite strain beam and rod theories and work done in nonlinear beam kinematics allowed for improved computational procedures in multibody dynamics [18–20, 30, 33, 52–54]. Downer *et al.* [20] presents one such method based on flexible beam finite elements, and achieves accurate representations of large deformations using a mesh of 8 to 12 beam elements. The code is applied to various beam maneuvers to obtain simulations. The authors credit the success to the "accurate computation of the nonlinear internal forces" [20], which earlier publications mention as an obstacle to modeling flexible systems [48]. More recently, a "hybrid" finite element technique was used by Hopkins and Ormiston [30] to model nonlinear deformation of a helicopter rotor blade. Nonlinear beam elements are formed with a moderate deformation theory, and the reference frame at the root of each element is constrained to move with the tip of its "parent" element. The authors achieve accurate representation of extremely large rotations (far more than the normal operation of a rotor blade) using a mesh of 20 nonlinear beam elements. These methods, like most today, apply numerical integration techniques to differential algebraic equations (DAEs) to obtain solutions.

2.3 DAE Methods

In general, there are two methods for formulating the equations of motion and constraints which are required to perform a dynamic analysis. One method is to generate the differential equations in a minimum coordinate set form, which includes only the independent coordinates of the system. This can be accomplished using constraint equations to eliminate dependent coordinates, or by using only the independent coordinates as generalized coordinates when deriving the equations. The major cost is incurred during the assembly process. The alternative is a maximum coordinate set form, which includes both dependent and independent coordinates as generalized coordinates. In this formulation, the constraint equations are appended to the differential equations of motion using Lagrange multipliers, which produces a set of DAEs [38].

Differential equations of motion are formed for each component of the system, algebraic equations are obtained from each joint, and the set is assembled for the entire system. Because DAEs constitute a stiff set of equations, they require robust algorithms to solve. Research in the late 1980s investigated the efficiency, stability, and robustness of various methods for solving these equations [42–44]. Many commercial software codes used in industry, such as ADAMS [49] (Mechanical Dynamics, Inc.) and DADS [24] (Computer Aided Design Software, Inc.), use DAE solvers to perform numerical integration.

2.4 Hamilton's Law

One common criticism of DAE solvers is that they are "not sufficiently robust and may fail to produce a solution for some configurations" [38]. It is not the objective of this research to investigate the use or implementation of a better DAE solver, but instead to take an entirely different approach to the problem. While most DAE solvers begin by deriving differential equations of motion from Lagrange's equation [1, 50], the current research begins with Hamilton's Law of varying action:

$$\int_{t_1}^{t_2} (\delta L + \overline{\delta W})\, dt - \sum_{k=1}^{n} \frac{\partial T}{\partial \dot{q}_k} \delta q_k \Big|_{t_1}^{t_2} = 0 \tag{2.1}$$

Here L is the Lagrangian, δW is the virtual work from external forces, and the generalized coordinates are q_k. Using an approach which is described in detail in chapter 3, equations for a beam are derived from a weak form of this equation and reduced to a set of algebraic equations using finite element techniques. This eliminates the need to numerically integrate a system of DAEs. Instead, the problem appears as a system of nonlinear algebraic equations.

Bailey [5, 6] was the first to demonstrate that Hamilton's Law can be used to obtain direct solutions to non-stationary, non-conservative, initial value problems. He proved that this can be accomplished "without any reference to or knowledge of differential equations" [6]. Substituting a basic truncated power series for q, he integrated

equation (2.1) to obtain a system of algebraic equations. Using this technique, he obtained solutions for various dynamics problems, including particle motion [5] and problems with rigid bodies [7]. Applying linear constitutive and kinematic equations, Bailey further extended this concept to problems of beam vibrations [8]. He achieved accurate results using second and third order polynomials as shape functions.

Following the initial work by Bailey, other authors began using Hamilton's Law to solve problems in dynamics. Riff and Baruch [9] derived $6n$ formulations of Hamilton's Law (for n degrees of freedom) based on various combinations of initial and final conditions for the state variations (δq_k). They used this technique to obtain solutions to simple mass-spring systems. Borri *et al.* [14] applied a standard finite element approximation to (2.1) with piecewise linear shape functions for the generalized coordinates. The authors used the model to analyze the response and stability of nonlinear periodic systems. Later work [15, 29] demonstrated that a mixed finite element formulation (where generalized coordinates and their momenta appear as independent unknowns) yields solutions with unconditional stability without having to use reduced element quadrature. The current research is also based on a mixed finite element formulation.

While time marching techniques are commonly applied to initial value problems, finite elements in time are also used in the literature [46, 47, 51] to discretize Hamilton's Law and obtain approximate solutions to simple dynamic systems. Interpolation polynomials are used to approximate the field quantities between time steps. Otherwise, the procedure is similar to a time marching algorithm applied to an initial value problem. Riff and Baruch [47] use third-order interpolation polynomials for displacements and first-order polynomials for the variation. The field variables and their variations are mutually independent, and appear as separate variables (a mixed formulation).

Time finite element discretizations of Hamilton's Law work well for particle and rigid body motion, where only integration in time is necessary. However, flexible

bodies also require spatial integration over their domain. A finite element model for such a problem requires interpolation polynomials for both the space and time domain. The result is a simultaneous boundary value problem in space and initial value problem in time. The boundary value problem determines the configuration of the body in space, and the initial value problem propagates this configuration through each time step. This requires the use of "space-time" finite element techniques.

2.5 Finite Elements in Space and Time

Early concepts of space-time finite elements were developed by Oden [41], Fried [22], and Argyris and Scharpf [2]. These works are based on Hamilton's Principle, which is a variation of Hamilton's Law. The same principles used to discretize the spatial domain for boundary value problems are applied to the time domain, and the two domains are treated equally to achieve time accurate simulations. Simultaneously discretizing the space and time domain sets this method apart from previously used semi-discretization techniques, where the problem is discretized into finite elements in space, reduced to a system of ordinary differential equations in time, and marched in time with finite difference methods [3, 34].

Bilinear formulations are often used in the literature to construct space-time finite element models [31, 45]. It allows for a consistent treatment of field quantities across both space and time domains. Peters and Izadpanah [45] suggest that treating space and time equally offers a unified solution strategy that will improve computational efficiency. Further research [31] illustrates that discretizing Hamilton's Law in this manner allows the user to benefit from the analogy between the "action" in the time domain and the energy in the space domain.

Several space-time finite element discretizations appear in the literature for beam and rod applications. Grohmann *et al.* [23] use a time-discontinuous Galerkin formulation for Timoshenko beam problems. The field quantities are continuous across boundaries of space but discontinuous between time "slabs". In other words, continuity is satisfied between elements at any given time, but not guaranteed between

consecutive time steps for any particular element. Atilgan and Hodges [3] apply a mixed formulation to rods, beginning with the governing differential equations and integrating by parts. This allows the use of piecewise constant shape functions for field variables and bilinear functions for variational quantities. Integration produces a system of algebraic equations which are applied to a structured mesh of rectangular space-time elements. Hodges [28] presents an energy preserving algorithm for space-time finite elements in beams. Additional studies use unstructured space-time meshes [34] and triangular elements [3, 31] to solve problems of wave propagation in rods.

2.6 Hamilton's Principle

There are many techniques for developing finite element models. In this research, the space-time finite element model is derived from Hamilton's Law. Before proceeding further one must understand the difference between various forms of "Hamilton's Law" in the literature. The trailing terms in equation (2.1) represent the variation of the generalized coordinates at the beginning and end of the time interval. Wherever a quantity is known, its variation will be zero. For an initial value problem (which is precisely what we are trying to solve), the trailing terms in the equation are unknown at t_2, but are zero for t_1. In the literature, however, it is common to enforce a stationarity requirement at the endpoints, making the variation zero and the trailing terms vanish. This results in Hamilton's Principle:

$$\int_{t_1}^{t_2} (\delta L + \overline{\delta W})\, dt = 0 \tag{2.2}$$

Hamilton's Principle is well established as a means for developing differential equations of motion for dynamic systems [1, 21, 40, 55]. In many MSA software programs, it is the basis for deriving the equations of motion, and numerical techniques are used to solve the resulting set of DAEs [49]. Bauchau *et al.* [11] develops an algorithm for flexible body MSA where differential equations of motion are derived from

Hamilton's Principle and virtual work for various types of bodies (cables, beams, and shells).

While useful for determining the differential equations for a system (for a DAE solver), Hamilton's Principle cannot be used to determine direct solutions to initial value problems because of the way it is derived. Unless the answer is known in advance, it is impossible to set up the problem such that the variation at t_2 is zero [6,9]. Therefore, the trailing terms of equation (2.1) are necessary for accurate numerical simulations to initial value problems and must be included.

Several authors point out that the incorrect use of the trailing terms in Hamilton's Law will result in convergence problems or incorrect solutions. Borri [14] notes that the trailing terms should be written as momenta (P) and not explicitly as ($\partial L/\partial \dot{q}$). A mixed formulation makes this possible. Peters and Izadpanah [45] define these terms as the virtual action leaving and entering the system at the space time boundaries. Virtual action is the time integral of virtual work at the spatial boundaries, and the spatial integral of "virtual action density" ($P\delta u$) on the time boundaries. Failure to account for these terms is analogous to neglecting energy entering and leaving the system while trying to enforce conservation laws.

2.7 Hamilton's Weak Principle

Hamilton's Weak Principle is a derivation of Hamilton's Law (not Hamilton's Principle) that offers several advantages for initial value problems. Start with the variation of the Hamiltonian (H), which is defined as follows [56]:

$$H(q, P, t) \equiv P^T \dot{q} - L(q, \dot{q}, t) \tag{2.3}$$

Evaluate the variation of the Hamiltonian and rearrange to solve for the variation in the Lagrangian (δL). Substitute δL back into equation (2.1), and integrate by parts

to eliminate $\dot{q}$ and obtain the following relationship:

$$\int_{t_1}^{t_2} (\delta\dot{q}^T P - \delta\dot{P}^T q - \delta H + \delta q^T Q)dt - \delta q^T P|_{t_1}^{t_2} + \delta P^T q|_{t_1}^{t_2} = 0 \tag{2.4}$$

Here Hamilton's Law is expressed in the "weak" form, and is thus called Hamilton's Weak Principle.

Using HWP as the mechanism for deriving the equations of motion offers many unique advantages which are applied in the literature. First, HWP generates a mixed formulation where generalized coordinates and momenta are independent field variables. This formulation is well suited for finite element approximations, because it is geometrically exact and provides a more accurate solution for a given level of computational power than a displacement formulation [26].

Second, HWP eliminates all derivatives of field quantities from the equation using integration by parts. Because there are no derivatives, only simple shape functions are necessary to satisfy C^0 continuity between elements. Higher order elements (p elements) could also be developed; however, for general nonlinear problems the use of crude shape functions is more efficient in that it allows element quadrature to be performed by inspection [29]. Independent variables need only be piecewise continuous, while variational quantities must be continuous and piecewise differentiable. The variational quantities will eventually drop from the equation completely.

Third, geometric constraints are included in the equation with Lagrange multipliers and the natural boundary conditions are contained explicitly in the equation (they are the trailing terms). Therefore all of the information about the system is included in a single equation. After integration and separation by variational coefficients, the result is a system of algebraic equations. For the present work, these equations are applied to space-time finite elements. For rigid bodies, they may be solved with a time marching algorithm.

The literature shows that HWP is an effective alternative for generating numerical solutions to problems of dynamics. A time marching procedure was used in [45] to

demonstrate accuracy for some simple problems. Hodges [29] applies HWP in mixed variational form to optimal control problems and nonlinear initial value problems. He also uses the technique to develop exact intrinsic equations for a moving beam [26], a derivation which is used extensively in this research.

Hamilton's Weak Principle is suggested [38] as a unifying theoretical basis for MSA that includes both rigid and elastic bodies. It has been demonstrated successfully in simulations of nonlinear rigid body motion such as a planar double pendulum and ballistic trajectories, taking full advantage of the properties listed above and achieving results comparable in accuracy to ODE solutions [39]. The equations of motion are developed with HWP, constraints are appended with Lagrange multipliers, and the system is discretized with simple shape functions to generate fully algebraic equations with displacements and momenta as independent variables. The same technique is used in this research, only additional field quantities such as strains and internal forces appear in order to account for flexible body motion.

2.8 Context of the Current Work

The goals of the current research are to develop a space-time finite element model using HWP and achieve time accurate simulations of a moving, elastic beam. This is inherently an initial value problem in time and a boundary value problem in space, containing differential equations of motion and algebraic constraints. Many approaches have been used over the last 30 years to solve this type of problem. The goal is to achieve success using a different method.

Most MSA programs are based on differential equation theory and use numerical integration with DAE solvers. Much research has gone into improving the accuracy and robustness of these methods. Yet they are still not sufficiently robust for certain types of aerospace applications. An alternate approach is offered through HWP which has not yet been fully explored. By using energy (or action) principles rather than differential equations, and integrating by parts to eliminate derivatives and "weaken" the problem, HWP allows an approach where DAEs do not appear. While it has

been used for various finite element techniques and has been proven to obtain direct solutions to rigid body motion, time accurate simulations of flexible beam problems using HWP have not yet been performed. This may offer a unified theory for MSA programs.

III. Theory

The purpose of this chapter is to provide the reader with a summary of the theory used in obtaining solutions for moving, elastic beam problems. The first half focuses on the theory used to derive the model, and the second deals with the algorithm for solving the equations. The basic geometry, reference frames, conventions, and assumptions are established first. Next, the kinematic and constitutive equations of the beam are written. Energy expressions are derived for kinetic energy, potential energy, and the virtual work due to external forces. The energy expressions are substituted directly into Hamilton's Law and the kinematic equations are appended to it with Lagrange multipliers. The resulting equation is discretized in space and time into a mixed finite element model, which produces a system of nonlinear algebraic equations. These equations are solved numerically using unconstrained optimization techniques to obtain solutions for various classes of problems.

3.1 Conventions

All vector quantities are designated by a subscript which indicates the basis of that vector. Superscripts in front of derivatives indicate which frame the derivative is taken in. The absence of a superscript indicates that the derivative is taken in the vector's reference frame. The cross product operator ($\tilde{}$) is used to represent a vector whose measure numbers are placed into a skew symmetric matrix as follows:

$$\tilde{A} = \begin{bmatrix} 0 & -A_3 & A_2 \\ A_3 & 0 & -A_1 \\ -A_2 & A_1 & 0 \end{bmatrix} \tag{3.1}$$

Extensive use of this operator appears in the derivation, including the following properties:

$$\tilde{A}B = -B\tilde{A} \tag{3.2}$$

$$\tilde{A}\tilde{B} = \tilde{B}\tilde{A} + \widetilde{\tilde{A}B} \tag{3.3}$$

A vector denoted with a hat ($\hat{q}$) represents a quantity that is defined at a discrete boundary of an element, while a bar ($\overline{q}$) represents a quantity that is defined within the domain of the element. The operator (δ) represents the variation of a function, except when it is written with a bar ($\overline{\delta q}$), where it indicates a virtual quantity.

3.2 Assumptions

In [26], Hodges uses HWP to develop the exact, intrinsic equations for a moving beam. This research uses the same process for developing the beam equation. The beam and its properties are generalized to a one-dimensional reference line. For thin beams with closed cross sections and moderate curvature, a simple 2-D cross-sectional analysis will determine the constitutive properties [4]. The advantage of this approach is that equations may be written for a beam of arbitrary cross sectional properties along its length.

The equations developed here are valid for beams with closed cross sections where warping is unrestrained. The in- and out-of-plane warping displacements do not need to be considered explicitly in the one dimensional analysis, but their effects are included implicitly in the structural equations [26,28]. Warping displacements are small and thus ignored in developing the kinematic equations.

It is assumed that a constitutive law may be written in terms of generalized strains alone. The strain energy per unit length depends on geometric and material properties. It is shown by Danielson and Hodges [18] to be a nonlinear function of strains, which are in turn nonlinear functions of displacements. Geometric stiffness due to this nonlinearity must be handled separately (*i.e.* open cross sections and trapeze effect).

3.3 Coordinate Reference Frames

Three different reference frames (or bases) for vectors are used in the analysis: A, B, and b. The A frame is known, and its motion relative to an inertial frame is

also known. For this research the A frame is treated as an inertial frame to eliminate excessive coordinate transformations. However, any number of arbitrary frames can be added provided that A can be traced back through them to the inertial frame. The b frame is a coordinate system that moves with the undeformed beam, and the B frame is attached to beam after deformation.

Figure 3.1 shows the geometry of the beam with bases b and B attached to their respective cross sections. The direction cosine matrix which transforms vectors from bases b to B is C^{Bb}. This matrix is a function of the beam reference line, varying along the length of the beam due to its curvature. In the b reference frame, b_1 always points in the direction of the beam's longitudinal axis. The vectors b_2 and b_3 are orthogonal to each other and parallel to the face of the cross section. The basis vectors for B are similar, except that B_1 is not necessarily parallel to the deformed beam's axis. This vector is always normal to the deformed cross section at the reference line, which may not coincide with the reference line when warping is present.

3.4 Rotation Parameters

Weiner-Milenkovic parameters [12] are used to describe the angular deformation (twist and bending) that occurs between the b and B reference frames. These rotation parameters are chosen because the singularity exists at 2π, and this is unlikely to be encountered for problems of beam dynamics. The following relationships are used:

$$C^{Bb} = (H^{Bb})(H^{Bb})^{-T} \tag{3.4}$$

$$H^{Bb} = \frac{(2\bar{\rho}_{B/b} + \frac{1}{2}\rho_{B/b}^T\rho_{B/b})[I] - 2\tilde{\rho}_{B/b} + \frac{1}{2}\tilde{\rho}_{B/b}\tilde{\rho}_{B/b}}{(4 - \bar{\rho}_{B/b})^2} \tag{3.5}$$

$$\bar{\rho}_{B/b} = 2 - \frac{1}{8}\rho_{B/b}^T\rho_{B/b} \tag{3.6}$$

In these equations, H^{Bb} is the three-by-three rotation parameter matrix and is distinguishable from H_B (the angular momentum) by its superscripts. The rotation

parameters are the measure numbers of the vector $\rho_{B/b}$ and contain the angular deformation information.

For small angles and bending about one axis, accurate slope information is obtainable directly from the Weiner-Milenkovic parameters ($\rho_{B/b}$). However, for problems with large deflections and rotations these parameters must be converted to either Euler angles or Euler rotation parameters to get a geometrically correct interpretation of the slope (θ). Using the direction cosine matrix, the Euler angles are determined as follows:

$$\theta = \cos^{-1}(C_{33}) \tag{3.7}$$

$$\psi = \sin^{-1}\left(\frac{C_{31}}{\sin(\theta)}\right) \tag{3.8}$$

$$\phi = \sin^{-1}\left(\frac{C_{13}}{\sin(\theta)}\right) \tag{3.9}$$

The inverse cosine function will only return a value between 0 and π (first and second quadrants). For deformations greater than π, the function will return an incorrect value. One fix is to subtract this value from 2π whenever the rotation is known to be greater than 180 degrees. The other option is to use Euler rotation parameters:

$$\phi = 4\tan^{-1}\left(\frac{1}{4}\sqrt{\rho(1)^2 + \rho(2)^2 + \rho(3)^2}\right) \tag{3.10}$$

$$u(1) = \frac{\rho(1)}{4\tan\left(\frac{\phi}{4}\right)} \tag{3.11}$$

$$u(2) = \frac{\rho(2)}{4\tan\left(\frac{\phi}{4}\right)} \tag{3.12}$$

$$u(3) = \frac{\rho(3)}{4\tan\left(\frac{\phi}{4}\right)} \tag{3.13}$$

Here, ϕ gives the rotation angle about the vector u, which is parallel to the axis of rotation for bending about a single axis. The inverse tangent function returns a value between $-\pi/2$ and $\pi/2$, but when this information is combined with the direction of u the complete slope information is available.

Hodges derives a space-time finite element algorithm for beams where neither displacement nor rotation variables appear, using generalized forces, moments, and strains [28]. The advantage of this approach is that the maximum order of nonlinearity is reduced to two. If displacement and rotation variables appear in the formulation, there is no limit to the degree of nonlinearity in the solution unless Euler parameters or the direction cosines are used as variables. However, this adds more unknowns to the problem, and therefore more Lagrange multipliers are necessary in the formulation. This approach results in a mixed formulation that is very similar to the one derived in the current research. Eliminating rotation parameters entirely or using the direction cosines directly as variables are both alternative approaches that are not used here, but are possible.

3.5 Beam Geometry

The geometry of the beam before and after deformation is shown in Figure 3.1. The reference line r follows the undeformed beam's longitudinal axis, where x_1 is the length along this line to the point of interest. Likewise, R is the same reference line after deformation, and the length along this line to the same material point is s. The vector $\bar{r}$ denotes the position of a point on the reference line of the undeformed beam, and it is a function of x_1, while $\bar{R}$ is the position vector of the same point after deformation and is a function of s. The displacement vector $\bar{u}$ is the difference between these two vectors, giving the total displacement (translation) of the section. Twisting and bending deformations are captured in the direction cosine matrix (C^{Bb}).

Let r_A^* represent the position of an arbitrary material point in the undeformed beam, and R_A^* be the location of the same material point after deformation. The

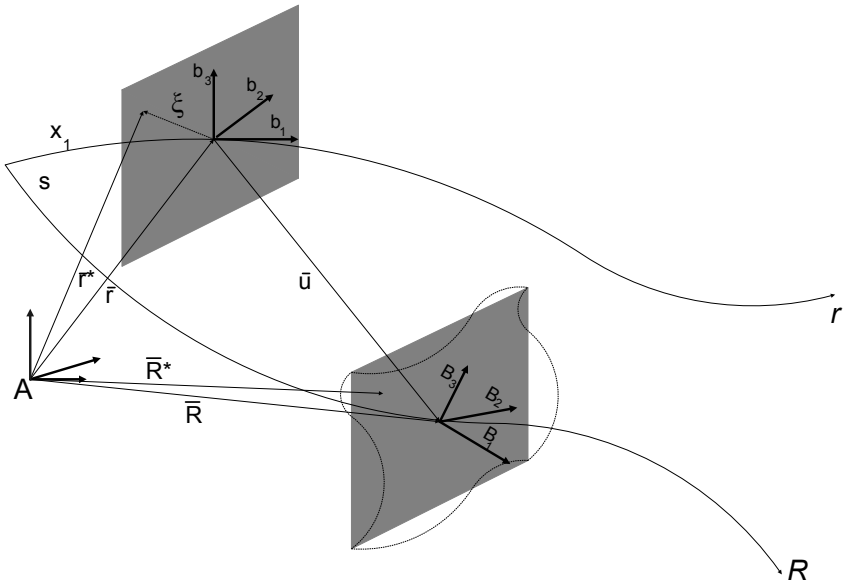


Figure 3.1: Geometry Schematic

following expressions are used to define the position in the deformed basis [26]:

$$r_b^* = r_b + \xi_b = r_b + x_2 b_2 + x_3 b_3 \tag{3.14}$$

$$R_B = C^{Bb}(r_b + u_b) \tag{3.15}$$

$$R_B^* = R_B + C^{Bb}(\xi_b + w_b) = R_B + x_2 B_2 + x_3 B_3 + w_i B_i \tag{3.16}$$

In these equations, ξ_b gives the position of a material point in the beam with respect to the reference line. It is always parallel to the cross section, and is of the form $[0, x_2, x_3]$. Cross-sectional warping is designated by $w_i B_i$. Any point in the undeformed beam is reached by moving on the reference line to the cross section (r_b), then along u_b to the deformed reference line, then through ξ_b to the point in the cross section, then along w_b to account for warping, and finally transforming to the B frame with C^{Bb}. While it is necessary to determine the warping in order to pinpoint the location of any material point in the beam after deformation, this is trivial to the analysis, which only requires knowledge of the point on the reference line and the properties of the cross section.

3.6 Beam Kinematics

Using the geometry defined in the previous section, the kinematics of the beam are now developed to establish strain-displacement and strain-velocity relationships. Begin by writing an expression for the inertial velocity of a material point in the beam:

$$ {}^{I}\dot{R}^{*}_{B} = {}^{I}\,\dot{r}_{B} + {}^{I}\,\dot{u}_{B} + {}^{I}\,\dot{\xi}_{B} + {}^{I}\,\dot{w}_{B} \tag{3.17} $$

The third and fourth terms are due to warping and are negligible in the analysis [26]. The first two terms will be evaluated further to determine an expression for the velocity of the beam in its deformed basis (V_B).

$$ \begin{aligned} V_B &= C^{Bb}({}^{I}\dot{R}^{*}_{b}) \\ V_B &= C^{Bb}({}^{I}\dot{r}_{b} + {}^{I}\,\dot{u}_{b}) \\ V_B &= C^{Bb}({}^{b}\dot{r}_{b} + \omega_b \times r_b + {}^{b}\,\dot{u}_{b} + \omega_b \times u_b) \\ V_B &= C^{Bb}(v_b + {}^{b}\,\dot{u}_{b} + \tilde{w}_b u_b) \end{aligned} \tag{3.18} $$

Here, v_b is the inertial velocity of a point in the undeformed reference frame. It is a known quantity that is prescribed by the user in the algorithm. Likewise, ω_b is the inertial angular velocity of the undeformed beam, and is also prescribed in the problem. The angular velocity of the deformed beam (Ω_B) is determined using the inertial angular velocity of the undeformed beam (ω_b) and the time derivative of the rotation matrix [26, 37]:

$$ \begin{aligned} \tilde{\Omega}_B &= -\dot{C}^{BA}C^{AB} \\ \tilde{\omega}_b &= -\dot{C}^{bA}C^{Ab} \\ \tilde{\Omega}_B &= -\dot{C}^{Bb}C^{bB} + C^{Bb}\tilde{\omega}_b C^{bB} \end{aligned} \tag{3.19} $$

Equations (3.18) and (3.19) are used to completely describe the velocity of the beam in the deformed frame. They include deformations due to "rigid body" motion and

the additional motion due to strain. The angular velocity is also written in terms of rotation parameters:

$$\Omega_B = H^{Bb}\dot{\rho}_{B/b} + C^{Bb}\omega_b \tag{3.20}$$

Next, the strain-displacement relationships are developed. The engineering strain is defined in [18], using generalized force and moment strains. Force strains, due to axial and shear forces, are represented by γ. Moment strains, due to torsion and bending, are designated with κ. The curvature of the deformed beam is K_B, and the curvature of the undeformed beam is k_b.

$$\gamma_b = C^{bB}R'_B - r'_b \tag{3.21}$$

$$\kappa_b = C^{bB}K_B - k_b \tag{3.22}$$

Here the $()'$ represents a derivative with respect to x_1, and is taken in the A frame. The curvatures can be found with the following relationships [18]:

$$\begin{aligned}
\tilde{K}_B &= -(C^{BA})'C^{AB} \\
\tilde{k}_b &= -(C^{BA})'C^{AB} \\
\tilde{K}_B &= -(C^{Bb})'C^{bB} + C^{Bb}\tilde{k}_b C^{bB}
\end{aligned} \tag{3.23}$$

Because curvatures are defined as the change in twist (K_{B1}) or slope (K_{B2}, K_{B3}) per unit length, their definitions come from spatial derivatives of the direction cosine matrices, whereas the angular velocities (which are analogous) come from the time derivatives of these matrices. Note the similarity between (3.19) and (3.23).

For force strains, the tangent vector of the deformed beam is transformed into undeformed coordinates and the original tangent vector is subtracted from it. The moment strain is determined by subtracting the curvature of the undeformed beam from that of the deformed beam. Hodges [26] presents several forms of these strains,

the simplest being:

$$\gamma_B = C^{Bb}(e_1 + u_b' + \tilde{k}_b u_b) - e_1 \tag{3.24}$$

$$\kappa_B = K_B - k_b \tag{3.25}$$

where e_1 is the unit vector $[1, 0, 0]$. These are the definitions that will be used for the generalized strains throughout this document. The curvature can also be written in terms of rotation parameters:

$$K_B = H^{Bb}(\rho_{B/b})' + C^{Bb}k_b \tag{3.26}$$

3.7 Virtual Displacements and Rotations

Equations of motion are intrinsic when no displacement or orientation variables appear in them [27]. Instead, generalized coordinates are used. Let q represent a generalized displacement, and ψ represent a generalized rotation. It is also possible to derive kinematics and dynamics from Hamilton's Law without introducing specific rotational coordinates (orientation angles), so that the user may choose which parameters to use. In order to proceed further, it is necessary to define virtual displacements and rotations. Let the virtual displacement be defined as the variation in the displacement:

$$\overline{\delta q} = \delta u_b \tag{3.27}$$

When the δq appears with a bar, it is used to represent a virtual quantity and not an operator associated the variation of a function. Likewise, the virtual rotation ($\delta\psi$) is defined as [37]:

$$\widetilde{\overline{\delta\psi}}_B = -\delta C^{Bb} C^{bB} \tag{3.28}$$

In the following section these definitions will be used to develop generalized speeds and strains and their variations.

3.8 Generalized Speeds and Strains

The purpose of this section is to develop expressions for the variation in generalized speeds and strains. Note that the variation in K_B is equal to the variation in κ_B since the undeformed curvature is known and its variation is zero. Therefore, most calculations will involve the use of K_B in lieu of κ, and for problems with initial curvature the moment strain may be determined using equation (3.25). Begin by taking the variation of the speeds and strains using Kirchoff's Kinetic Analogy:

$$\delta\tilde{K}_B = -\delta(C^{Bb})'C^{bB} - (C^{Bb})'\delta C^{bB} + \delta C^{Bb}\tilde{k}_b C^{bB} + C^{Bb}\tilde{k}_b\delta C^{bB} \tag{3.29}$$

$$\delta\gamma_B = \delta C^{Bb}(e_1 + u_b' + \tilde{k}_b u_b) + C^{Bb}(\delta u_b' + \tilde{k}_b\delta u_b) \tag{3.30}$$

$$\delta\tilde{\Omega}_B = -\delta\dot{C}^{Bb}C^{bB} - \dot{C}^{Bb}\delta C^{bB} + \delta C^{Bb}\tilde{\omega}_b C^{bB} + C^{Bb}\tilde{\omega}_b\delta C^{bB} \tag{3.31}$$

$$\delta V_B = \delta C^{Bb}(v_b + \dot{u}_b + \tilde{\omega}_b u_b) + C^{Bb}(\delta\dot{u}_b + \tilde{\omega}_b\delta u_b) \tag{3.32}$$

Next, take the derivatives of the virtual displacements and rotations in space and time:

$$\widetilde{\overline{\delta\psi}}'_B = -\delta(C^{Bb})'C^{bB} - \delta C^{Bb}(C^{bB})' \tag{3.33}$$

$$\dot{\widetilde{\overline{\delta\psi}}}_B = -\delta\dot{C}^{Bb}C^{bB} - \delta C^{Bb}\dot{C}^{bB} \tag{3.34}$$

$$\overline{\delta q}'_B = (C^{Bb})'\delta u_b + C^{Bb}\delta u_b' \tag{3.35}$$

$$\dot{\overline{\delta q}}_B = \dot{C}^{Bb}\delta u_b + C^{Bb}\delta\dot{u}_b \tag{3.36}$$

Rearrange (3.33) and (3.34) to get the following:

$$\delta(C^{Bb})'C^{bB} = \widetilde{\overline{\delta\psi}}'_B + \delta C^{Bb}(C^{bB})' \tag{3.37}$$

$$\delta\dot{C}^{Bb}C^{bB} = -\dot{\widetilde{\overline{\delta\psi}}}_B - \delta C^{Bb}\dot{C}^{bB} \tag{3.38}$$

Use (3.37) and (3.38) to eliminate the first terms from (3.29) and (3.31):

$$\delta\tilde{K}_B = \widetilde{\overline{\delta\psi}}'_B + \delta C^{Bb}(C^{bB})' - (C^{Bb})'\delta C^{bB} + \delta C^{Bb}\tilde{k}_b C^{bB} + C^{Bb}\tilde{k}_b\delta C^{bB} \tag{3.39}$$

$$\delta\tilde{\Omega}_B = \dot{\widetilde{\overline{\delta\psi}}}_B + \delta C^{Bb}\dot{C}^{bB} - \dot{C}^{Bb}\delta C^{bB} + \delta C^{Bb}\tilde{\omega}_b C^{bB} + C^{Bb}\tilde{\omega}_b\delta C^{bB} \tag{3.40}$$

Using the definitions of curvature (3.23) and angular velocity (3.19), rearrange to get the derivatives of the rotation matrix C^{Bb}:

$$(C^{Bb})' = C^{Bb}\tilde{k}_b - \tilde{K}_B C^{Bb} \tag{3.41}$$

$$\dot{C}^{Bb} = C^{Bb}\tilde{\omega}_b - \tilde{\Omega}_B C^{Bb} \tag{3.42}$$

Use these relationships to eliminate derivatives of the direction cosine matrix from (3.39) and (3.40):

$$\begin{aligned}\delta\tilde{K}_B = {} & \widetilde{\overline{\delta\psi}}'_B + \delta C^{Bb}(C^{Bb}\tilde{k}_b - \tilde{K}_B C^{Bb}) - (C^{Bb}\tilde{k}_b - \tilde{K}_B C^{Bb})\delta C^{bB} \\ & + \delta C^{Bb}\tilde{k}_b C^{bB} + C^{Bb}\tilde{k}_b\delta C^{bB}\end{aligned} \tag{3.43}$$

$$\begin{aligned}\delta\tilde{\Omega}_B = {} & \dot{\widetilde{\overline{\delta\psi}}}_B + \delta C^{Bb}(C^{Bb}\tilde{\omega}_b - \tilde{\Omega}_B C^{Bb}) - (C^{Bb}\tilde{\omega}_b - \tilde{\Omega}_B C^{Bb})\delta C^{bB} \\ & + \delta C^{Bb}\tilde{\omega}_b C^{bB} + C^{Bb}\tilde{\omega}_b\delta C^{bB}\end{aligned} \tag{3.44}$$

From the definition of virtual rotations:

$$\delta C^{Bb} = -\widetilde{\overline{\delta\psi}}_B C^{Bb} \tag{3.45}$$

$$\delta C^{bB} = C^{bB}\widetilde{\overline{\delta\psi}}_B \tag{3.46}$$

When substituted into (3.43) and (3.44), variations in the direction cosine matrices are eliminated:

$$\begin{aligned}\delta\tilde{K}_B &= \widetilde{\overline{\delta\psi}}'_B + \widetilde{\overline{\delta\psi}}_B C^{Bb}\tilde{k}_b C^{bB} - \widetilde{\overline{\delta\psi}}_B C^{Bb}C^{bB}\tilde{K}_B - C^{Bb}\tilde{k}_b C^{bB}\widetilde{\overline{\delta\psi}} \\ &\quad + \tilde{K}_B C^{Bb}C^{bB}\widetilde{\overline{\delta\psi}}_B - \widetilde{\overline{\delta\psi}}_B C^{Bb}\tilde{k}_b C^{bB} + C^{Bb}\tilde{k}_b C^{bB}\widetilde{\overline{\delta\psi}}_B \\ \delta\tilde{K}_B &= \widetilde{\overline{\delta\psi}}'_B - \widetilde{\overline{\delta\psi}}_B\tilde{K}_B + \tilde{K}_B\widetilde{\overline{\delta\psi}}_B \end{aligned} \tag{3.47}$$

Applying a similar procedure with $\delta\tilde{\Omega}_B$ yields the following equation:

$$\delta\tilde{\Omega}_B = \widetilde{\dot{\overline{\delta\psi}}}_B - \widetilde{\overline{\delta\psi}}_B\tilde{\Omega}_B + \tilde{\Omega}_B\widetilde{\overline{\delta\psi}}_B \tag{3.48}$$

Next, apply the cross product property defined in (3.3) to obtain:

$$\begin{aligned}\tilde{K}_B\widetilde{\overline{\delta\psi}}_B &= \widetilde{\overline{\delta\psi}}_B\tilde{K}_B + \widetilde{\tilde{K}_B\overline{\delta\psi}}_B \\ \tilde{\Omega}_B\widetilde{\overline{\delta\psi}}_B &= \widetilde{\overline{\delta\psi}}_B\tilde{\Omega}_B + \widetilde{\tilde{\Omega}_B\overline{\delta\psi}}_B \end{aligned}$$

This produces equations for the generalized angular speeds and moment strains:

$$\delta K_B = \overline{\delta\psi}'_B + \tilde{K}_B\overline{\delta\psi}_B \tag{3.49}$$

$$\delta\Omega_B = \overline{\dot{\delta\psi}}_B + \tilde{\Omega}_B\overline{\delta\psi}_B \tag{3.50}$$

Next, expressions are derived for generalized speeds and force strains. Rearranging (3.35) and (3.36) and replacing δu_b with $C^{bB}\overline{\delta q}_B$:

$$C^{Bb}\delta u'_b = \overline{\delta q}'_B - (C^{Bb})'C^{bB}\overline{\delta q}_B \tag{3.51}$$

$$C^{Bb}\delta\dot{u}^{\,b} = \overline{\dot{\delta q}}_B - \dot{C}^{Bb}C^{bB}\overline{\delta q}_B \tag{3.52}$$

The first term inside parenthesis in (3.32) is simply V_b, and the first term inside the parenthesis of (3.31) can be written as $(e_1 + \gamma_b)$. Replacing δC^{Bb} with its definition

from (3.45) and substituting (3.51) and (3.52), the following is developed:

$$\begin{aligned}
\delta V_B &= \delta C^{Bb} V_b + C^{Bb}(\delta \dot{u}_b + \tilde{\omega}_b \delta u_b) \\
\delta V_B &= -\widetilde{\overline{\delta\psi}}_B C^{Bb} V_b + \dot{\overline{\delta q}}_B - \dot{C}^{Bb} C^{bB} \overline{\delta q}_B + C^{Bb} \tilde{\omega}_b C^{bB} \overline{\delta q}_B \\
\delta \gamma_B &= \delta C^{Bb}(e_1 + u_b' + \tilde{k}_b u_b) + C^{Bb}(\delta u_b' + \tilde{k}_b \delta u_b) \\
\delta \gamma_B &= -\widetilde{\overline{\delta\psi}}_B (e_1 + \gamma_B) + \overline{\delta q}_B' - (C^{Bb})' C^{bB} \overline{\delta q}_B + C^{Bb} \tilde{k}_b C^{bB} \overline{\delta q}_B
\end{aligned} \tag{3.53}$$

Next replace $\dot{C}^{Bb}$ and $(C^{Bb})'$ with their definitions:

$$\begin{aligned}
\delta V_B &= \tilde{V}_B \overline{\delta\psi}_B + \dot{\overline{\delta q}}_B - (C^{Bb}\tilde{\omega}_b - \tilde{\Omega}_B C^{Bb}) C^{bB} \overline{\delta q}_B + C^{Bb} \tilde{\omega}_b C^{bB} \overline{\delta q}_B \\
\delta \gamma_B &= (\tilde{e}_1 + \tilde{\gamma}_B) \overline{\delta\psi}_B + \overline{\delta q}_B' - (C^{Bb}\tilde{k}_b - \tilde{K}_B C^{Bb}) C^{bB} \overline{\delta q}_B + C^{Bb} \tilde{k}_b C^{bB} \overline{\delta q}_B
\end{aligned} \tag{3.54}$$

Finally, simplify these expressions further to obtain the definitions of generalized speeds and force strains:

$$\delta V_B = \dot{\overline{\delta q}}_B + \tilde{V}_B \overline{\delta\psi}_B + \tilde{\Omega}_B \overline{\delta q}_B \tag{3.55}$$

$$\delta \gamma_B = \overline{\delta q}_B' + (\tilde{e}_1 + \tilde{\gamma}_B) \overline{\delta\psi}_B + \tilde{K}_B \overline{\delta q}_B \tag{3.56}$$

3.9 Kinetic and Potential Energy

Hamilton's Law requires the definition of kinetic and potential energy for the system. Let dT represent the kinetic energy of an infinitesimal area of a cross section. The kinetic energy (T) of a cross section of the beam (energy per unit length) may be defined as [38]:

$$T = \int_A dT = \frac{1}{2} m V_B^T V_B - m \Omega_B^T \tilde{V}_B \bar{\xi}_B + \frac{1}{2} \Omega_B^T I \Omega_B \tag{3.57}$$

Here, m represents mass per unit length and I is the mass moment of inertia matrix for the section of the beam. The location of the center of mass from the reference line is $\bar{\xi}_B$. In general, the reference line passes through the centroid and this term is zero.

In order to develop intrinsic equations using Hamilton's Law, the kinetic energy will be written as a function of generalized speeds, and its variation is evaluated using Kirchoff's Kinetic Analogy and the chain rule:

$$\delta T = \delta V_B^T \left(\frac{\partial T}{\partial V_B} \right)^T + \delta \Omega_B^T \left(\frac{\partial T}{\partial \Omega_B} \right)^T \tag{3.58}$$

Likewise, the potential energy per unit length (U) is defined using the concept of strain energy. The definition of strain energy is dependent on the constitutive equations of the solid, but in general is a function of the strains. Therefore, a definition can be written using the chain rule to get the variation:

$$\delta U = \delta \gamma_B^T \left(\frac{\partial U}{\partial \gamma_B} \right)^T + \delta \kappa_B^T \left(\frac{\partial U}{\partial \kappa_B} \right)^T \tag{3.59}$$

The values in parenthesis will remain in general form for the present, since they are dependent on the problem.

Gravity is a contributor to potential energy, although its contribution to the problem could be incorporated using virtual work. Introducing it here allows gravity to be turned on or off in the algorithm by specifying a gravitational vector (g_B). For scenarios with long, flexible beams under light loads, the influence of the beam's own weight can play a large factor. There are other scenarios where gravity should be neglected. For dynamic problems, the orientation of gravity relative to the undeformed beam must be known at all times.

The potential energy due to gravity is specified for an incremental mass in the cross section as:

$$dU_g = -g_B^T (R_B^* - R_{ref\ B}) dm \tag{3.60}$$

Integrating over the cross-sectional area gives the potential energy per unit length, and the variation is determined as well. Note that the variation of known quantities

is zero. Using Kirchoff's Kinetic Analogy:

$$
\begin{aligned}
U_g &= -g_B^T \int_A (R_B^* - R_{ref\ B}) dm \\
\delta U_g &= -g_B^T \int_A (\delta R_B^* - \delta R_{ref\ B}) dm \\
\delta U_g &= -g_B^T \int_A (\delta r_B + \delta u_B + \delta \xi_B) dm \\
\delta U_g &= -g_B^T \int_A (\delta u_B + \delta \xi_B) dm \\
\delta U_g &= -g_B^T \int_A (\delta q_B + \widetilde{\overline{\delta \psi}}_B \xi_B) dm \\
\delta U_g &= -g_B^T (\delta q_B - \tilde{\bar{\xi}}_B \overline{\delta \psi}_B) m
\end{aligned} \tag{3.61}
$$

Adding these terms has the same effect as adding the following distributed forces and moments to the beam in the direction of the gravity vector:

$$
\begin{aligned}
f_g &= m g_B \\
m_g &= \tilde{\bar{\xi}}_B m g_B
\end{aligned}
$$

Either approach will work. Note that $\bar{\xi}_B$ is defined in the deformed coordinate system. It is more realistic to use $C^{Bb}\bar{\xi}_b$, since the vector $\bar{\xi}_b$ is known (in the undeformed coordinate system). This is also true of the gravity vector, g_B, which must be given in the deformed coordinate system. The user will know g_b, and therefore in the equations it must be pulled forward into the deformed coordinate system using $C^{Bb} g_b$.

3.10 Generalized Momenta, Forces, and Moments

The generalized momenta (P_B and H_B) can be obtained from the kinetic energy through the following relationships [26]:

$$
\begin{aligned}
P_B &= \left(\frac{\partial T}{\partial V_B} \right) = m(V_B - \tilde{\bar{\xi}}_B \Omega_B) \\
H_B &= \left(\frac{\partial T}{\partial \Omega_B} \right) = I\Omega_B + m\tilde{\bar{\xi}}_B V_B
\end{aligned} \tag{3.62}
$$

Using these definitions, the kinetic energy can also be written as follows:

$$\delta T = \delta V_B^T P_B + \delta \Omega_B^T H_B \tag{3.63}$$

Note that the momenta impose a type of constitutive equation on the problem, relating velocities to momenta in the same way that strains will be related to internal forces. These equations may be assembled in the following matrix form:

$$\begin{bmatrix} P_B \\ H_B \end{bmatrix} = \begin{bmatrix} m & -m\tilde{\tilde{\xi}}_B \\ m\tilde{\tilde{\xi}}_B & I \end{bmatrix} \begin{bmatrix} V_B \\ \Omega_B \end{bmatrix} \tag{3.64}$$

The generalized forces (F_B) and moments (M_B) are formed in a similar fashion. For a known strain energy function (dependent on geometric and material properties), which is a function of γ_B and κ_B alone, the internal forces and moments may be determined by its partial derivatives:

$$\begin{aligned} F_B &= \left(\frac{\partial U}{\partial \gamma_B} \right) \\ M_B &= \left(\frac{\partial U}{\partial \kappa_B} \right) \end{aligned} \tag{3.65}$$

Note the similarity to the way kinetic energy relates to generalized momenta. This leads to the following definition of the variation in strain energy:

$$\delta U = \delta \gamma_B^T F_B + \delta \kappa_B^T M_B \tag{3.66}$$

The following set of generic equations are used to represent the constitutive laws:

$$\begin{bmatrix} F_B \\ M_B \end{bmatrix} = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix} \begin{bmatrix} \gamma_B \\ \kappa_B \end{bmatrix} \tag{3.67}$$

In this equation A, B, and D are three-by-three matrices containing geometric and material properties. For an isotropic beam, the following relationships are used for A

and D:

$$A = \begin{bmatrix} EA & 0 & 0 \\ 0 & GK_2 & 0 \\ 0 & 0 & GK_3 \end{bmatrix} \qquad D = \begin{bmatrix} GJ & 0 & 0 \\ 0 & EI_2 & 0 \\ 0 & 0 & EI_3 \end{bmatrix} \tag{3.68}$$

The matrix B is a three-by-three zero matrix for an isotropic solid. Geometric and material properties include Young's modulus of elasticity (E), shear modulus (G), torsional rigidity (J), cross sectional area (A), and effective shear areas (K). Although the theory allows for more complex beams, isotropic beams are used for the test cases in this research. These relationships hold true for Hookean materials while the solid remains in the elastic region of deformation.

3.11 Virtual Work and Virtual Action

The action of applied loads and moments is accounted for using the principle of virtual work. Let f_B and m_B represent externally applied forces and moments per unit length acting at the beam reference line. Since any load set can be written in terms of a force vector and a couple about the reference line of choice, this is sufficient for all possible cases. The virtual work due to applied loads (per unit length) can be written as [26]:

$$\overline{\delta W} = \overline{\delta q}_B^T f_B + \overline{\delta \psi}_B^T m_B \tag{3.69}$$

Externally applied loads appear in the system equations using this mechanism. The subscript shows that they are in the deformed coordinate system, indicating that they are follower forces. If desired, they can be pre-multiplied by a direction cosine matrix to stay in the undeformed coordinate system. As previously discussed, gravity could also be included as an external force.

The trailing terms in Hamilton's Law are sometimes referred to as virtual action [45]. They represent the natural spatial and temporal boundary conditions of

the problem. Hodges [26] shows that they can be expanded as:

$$\overline{\delta A} = \int_0^\ell (\overline{\delta q}_B^T \hat{P}_B + \overline{\delta \psi}_B^T \hat{H}_B)|_{t_1}^{t_2} dx_1 - \int_{t_1}^{t_2} (\overline{\delta q}_B^T \hat{F}_B + \overline{\delta \psi}_B^T \hat{M}_B)|_0^\ell dt \tag{3.70}$$

These conditions are enforced when deriving the beam equations from Hamilton's Law.

3.12 Derivation of Equations from Hamilton's Law

This section provides a detailed derivation of the exact, intrinsic equations for a beam beginning with Hamilton's Law. It is similar to the procedure performed by Hodges in [26]. Hamilton's Law was discussed in the previous chapter and is restated here:

$$\int_{t_1}^{t_2} (\delta L + \overline{\delta W})\, dt - \sum_{k=1}^{n} \frac{\partial T}{\partial \dot{q}_k} \delta q_k|_{t_1}^{t_2} = 0$$

First, rewrite the Lagrangian in terms of kinetic and potential energy, let the trailing terms be defined as the virtual action ($\overline{\delta A}$), and integrate over the length of the beam:

$$\int_{t_1}^{t_2} \int_0^\ell (\delta T - \delta U - \delta U_G + \overline{\delta W})\, dx_1 dt - \overline{\delta A} = 0 \tag{3.71}$$

Spatial integration is necessary because all of the energy terms previously defined have units of energy per length, and must be integrated over the domain. This integral is not necessary for rigid body or particle motion, but is required for deformable bodies as discussed in chapter 2. Insert the definitions of kinetic and potential energy (3.63 and 3.66) in terms of generalized speeds, strains, momenta, and forces:

$$\begin{aligned} \int_{t_1}^{t_2} \int_0^\ell [(\delta V_B^T P_B + \delta \Omega_B^T H_B) - (\delta \gamma_B^T F_B + \delta \kappa_B^T M_B) \\ - \delta U_G + \overline{\delta W})]\, dx_1 dt - \overline{\delta A} = 0 \end{aligned} \tag{3.72}$$

The next step requires introducing kinematic equations through the use of Lagrange multipliers (λ_n). The following equation contains the kinematic information, where

terms denoted with an asterisk will later be replaced with their definitions:

$$\delta \int_{t_1}^{t_2} \int_0^{\ell} \left[\lambda_1^T (V_B - V_B^*) + \lambda_2^T (\Omega_B - \Omega_B^*) + \lambda_3^T (\gamma_B - \gamma_B^*) + \lambda_4^T (\kappa_B - \kappa_B^*) \right] dx_1 dt = 0$$

Carry out the variation to obtain the following:

$$\begin{aligned} \int_{t_1}^{t_2} \int_0^{\ell} [& \delta\lambda_1^T (V_B - V_B^*) + \delta\lambda_2^T (\Omega_B - \Omega_B^*) \\ & + \delta\lambda_3^T (\gamma_B - \gamma_B^*) + \delta\lambda_4^T (\kappa_B - \kappa_B^*) \\ & + \lambda_1^T (\delta V_B - \delta V_B^*) + \lambda_2^T (\delta\Omega_B - \delta\Omega_B^*) \\ & + \lambda_3^T (\delta\gamma_B - \delta\gamma_B^*) + \lambda_4^T (\delta\kappa_B - \delta\kappa_B^*)] \, dx_1 dt = 0 \end{aligned} \tag{3.73}$$

Next, combine equations (3.73) and (3.72) and sort by like terms:

$$\begin{aligned} \int_{t_1}^{t_2} \int_0^{\ell} [& \delta V_B^T (P_B + \lambda_1) - \delta V_B^{T*} \lambda_1 + \delta\Omega_B^T (H_B + \lambda_2) - \delta\Omega_B^{T*} \lambda_2 \\ & + \delta\gamma_B^T (\lambda_3 - F_B) - \delta\gamma_B^{T*} \lambda_3 + \delta\kappa_B^T (\lambda_4 - M_B) - \delta\kappa_B^{T*} \lambda_4 \\ & + \delta\lambda_1^T (V_B - V_B^*) + \delta\lambda_2^T (\Omega_B - \Omega_B^*) + \delta\lambda_3^T (\gamma_B - \gamma_B^*) + \delta\lambda_4^T (\kappa_B - \kappa_B^*) \\ & - \delta U_G + \overline{\delta W}] \, dx_1 dt - \overline{\delta A} = 0 \end{aligned} \tag{3.74}$$

This equation yields the following relationships, which define the Lagrange multipliers:

$$\begin{aligned} \lambda_1 &= -P_B & \delta\lambda_1 &= -\delta P_B \\ \lambda_2 &= -H_B & \delta\lambda_2 &= -\delta H_B \\ \lambda_3 &= F_B & \delta\lambda_3 &= \delta F_B \\ \lambda_4 &= M_B & \delta\lambda_4 &= \delta M_B \end{aligned} \tag{3.75}$$

Equation (3.74) can thus be reduced to the following:

$$\begin{aligned}\int_{t_1}^{t_2}\int_0^{\ell}[&\delta V_B^{T*}P_B+\delta\Omega_B^{T*}H_B-\delta\gamma_B^{T*}F_B-\delta\kappa_B^{T*}M_B\\&+\delta V_B^T(P_B-P_B^*)+\delta\Omega_B^T(H_B-H_B^*)+\delta\gamma_B^T(F_B^*-F_B)+\delta\kappa_B^T(M_B^*-M_B)\\&-\delta P_B^T(V_B-V_B^*)-\delta H_B^T(\Omega_B-\Omega_B^*)+\delta F_B^T(\gamma_B-\gamma_B^*)+\delta M_B^T(\kappa_B-\kappa_B^*)\\&-\delta U_G+\overline{\delta W}]\,dx_1dt-\overline{\delta A}=0\end{aligned}\tag{3.76}$$

Next, insert the definitions of δV_B^*, $\delta\Omega_B^*$, $\delta\gamma_B^*$, $\delta\kappa_B^*$, P_B^*, H_B^*, F_B^*, and M_B^*, where $\delta K_B=\delta\kappa_B$ since the undeformed curvature is known.

$$\begin{aligned}\int_{t_1}^{t_2}\int_0^{\ell}[&(\dot{\overline{\delta q}}_B^T+\tilde{V}_B\overline{\delta\psi}_B^T+\tilde{\Omega}_B\overline{\delta q}_B^T)P_B+(\dot{\overline{\delta\psi}}_B^T+\tilde{\Omega}_B\overline{\delta\psi}_B^T)H_B\\&-((\overline{\delta q}_B^T)'+(\tilde{e}_1+\tilde{\gamma}_B)\overline{\delta\psi}_B^T+\tilde{K}_B\overline{\delta q}_B^T)F_B-((\overline{\delta\psi}_B^T)'+\tilde{K}_B\overline{\delta\psi}_B^T)M_B\\&+\delta V_B^T(P_B-mV_B+m\tilde{\bar{\xi}}_B\Omega_B)+\delta\Omega_B^T(H_B-I\Omega_B-m\tilde{\bar{\xi}}_BV_B)\\&+\delta\gamma_B^T(A\gamma_B+B\kappa_B-F_B)+\delta\kappa_B^T(B^T\gamma_B+D\kappa_B-M_B)\\&-\delta P_B^T(V_B-(C^{Bb}(v_b+{}^b\,\dot{u}_b+\tilde{w}_bu_b)))-\delta H_B^T(\Omega_B-(H^{Bb}\dot{\rho}_{B/b}+C^{Bb}\omega_b))\\&+\delta F_B^T(\gamma_B-(C^{Bb}(e_1+(u_b)'+\tilde{k}_bu_b)-e_1))\\&+\delta M_B^T(\kappa_B-(H^{Bb}(\rho_{B/b})'+C^{Bb}k_b-k_b))\\&-\delta U_G+\overline{\delta W}]dx_1dt-\overline{\delta A}=0\end{aligned}\tag{3.77}$$

Define the following bar quantities for generalized momenta and internal forces:

$$\begin{aligned}\overline{\delta P}&=C^{bB}\delta P_B & \overline{\delta F}&=C^{bB}\delta P_B\\ \delta P_B^T&=\overline{\delta P}^TC^{bB} & \delta F_B^T&=\overline{\delta F}^TC^{bB}\\ \overline{\delta H}&=(H^{Bb})^T\delta H_B & \overline{\delta M}&=(H^{Bb})^T\delta H_B\\ \delta H_B^T&=\overline{\delta H}^T(H^{Bb})^{-1} & \delta M_B^T&=\overline{\delta M}^T(H^{Bb})^{-1}\end{aligned}\tag{3.78}$$

Substitute these into equation (3.77) to obtain the following:

$$\int_{t_1}^{t_2}\int_0^{\ell}[(\dot{\overline{\delta q}}_B^T+\tilde{V}_B\overline{\delta\psi}_B^T+\tilde{\Omega}_B\overline{\delta q}_B^T)P_B+(\dot{\overline{\delta\psi}}_B^T+\tilde{\Omega}_B\overline{\delta\psi}_B^T)H_B$$
$$-((\overline{\delta q}_B^T)'+(\tilde{e}_1+\tilde{\gamma}_B)\overline{\delta\psi}_B^T+\tilde{K}_B\overline{\delta q}_B^T)F_B-((\overline{\delta\psi}_B^T)'+\tilde{K}_B\overline{\delta\psi}_B^T)M_B$$
$$+\delta V_B^T(P_B-mV_B+m\tilde{\bar{\xi}}_B\Omega_B)+\delta\Omega_B^T(H_B-I\Omega_B-m\tilde{\bar{\xi}}_BV_B)$$
$$+\delta\gamma_B^T(A\gamma_B+B\kappa_B-F_B)+\delta\kappa_B^T(B^T\gamma_B+D\kappa_B-M_B)$$
$$-(\overline{\delta P}^TC^{bB})V_B+(\overline{\delta P}^TC^{bB})(C^{Bb}(v_b+{}^b\,\dot{u}_b+\tilde{w}_bu_b))$$
$$-(\overline{\delta H}^T(H^{Bb})^{-1})\Omega_B+(\overline{\delta H}^T(H^{Bb})^{-1})(H^{Bb}\dot{\rho}_{B/b}+C^{Bb}\omega_b)$$
$$+(\overline{\delta F}^TC^{bB})\gamma_B-(\overline{\delta F}^TC^{bB})(C^{Bb}(e_1+u_b'+\tilde{k}_bu_b)-e_1)$$
$$+(\overline{\delta M}^T(H^{Bb})^{-1})\kappa_B-(\overline{\delta M}^T(H^{Bb})^{-1})(H^{Bb}(\rho_{B/b})'+C^{Bb}k_b-k_b)$$
$$-\delta U_G+\overline{\delta W}]\,dx_1dt-\overline{\delta A}=0 \tag{3.79}$$

From here the derivatives $\dot{u}_b$, $\dot{\rho}_{B/b}$, u_b', and $\rho_{B/b}'$ are evaluated using integration by parts. This will remove all derivatives of non-variational quantities from the problem while at the same time enforcing natural boundary conditions. This is the application of HWP in the problem. It will allow the choice of simple shape functions in the finite element integration as well. Performing integration by parts and expanding the terms

in the virtual action:

$$\begin{aligned}
\int_{t_1}^{t_2}\int_0^{\ell} & [(\dot{\overline{\delta q}}_B^T + \tilde{V}_B\overline{\delta\psi}_B^T + \tilde{\Omega}_B\overline{\delta q}_B^T)P_B + (\dot{\overline{\delta\psi}}_B^T + \tilde{\Omega}_B\overline{\delta\psi}_B^T)H_B \\
& - ((\overline{\delta q}_B^T)' + (\tilde{e}_1+\tilde{\gamma}_B)\overline{\delta\psi}_B^T + \tilde{K}_B\overline{\delta q}_B^T)F_B - ((\overline{\delta\psi}_B^T)' + \tilde{K}_B\overline{\delta\psi}_B^T)M_B \\
& + \delta V_B^T(P_B - mV_B + m\tilde{\bar{\xi}}_B\Omega_B) + \delta\Omega_B^T(H_B - I\Omega_B - m\tilde{\bar{\xi}}_BV_B) \\
& + \delta\gamma_B^T(A\gamma_B + B\kappa_B - F_B) + \delta\kappa_B^T(B^T\gamma_B + D\kappa_B - M_B) \\
& + \overline{\delta P}^T(v_b + \tilde{w}_bu_b - C^{bB}V_B) + \overline{\delta F}^T(C^{bB}(\gamma_B + e_1) - (e_1 + \tilde{k}_bu_b)) \\
& + \overline{\delta H}^T(H^{Bb})^{-1}(C^{Bb}\omega_b - \Omega_B) + \overline{\delta M}^T(H^{Bb})^{-1}(\kappa_B - C^{Bb}k_b + k_b) \\
& - \dot{\overline{\delta P}}^Tu_b + (\overline{\delta F}^T)'u_b - \dot{\overline{\delta H}}^T\rho_{B/b} + (\overline{\delta M}^T)'\rho_{B/b} \\
& - \delta U_G + \overline{\delta W}]\,dx_1dt \\
& - \int_{t_1}^{t_2}(\overline{\delta F}^T\hat{u}_b + \overline{\delta M}^T\hat{\rho}_{B/b})|_0^{\ell}dt \\
& + \int_0^{\ell}(\overline{\delta P}^T\hat{u}_b + \overline{\delta H}^T\hat{\rho}_{B/b})|_{t_1}^{t_2}dx_1 \\
& - \int_0^{\ell}(\overline{\delta q}_B^T\hat{P}_B + \overline{\delta\psi}_B^T\hat{H}_B)|_{t_1}^{t_2}dx_1 - \int_{t_1}^{t_2}(\overline{\delta q}_B^T\hat{F}_B + \overline{\delta\psi}_B^T\hat{M}_B)|_0^{\ell}dt = 0 \qquad (3.80)
\end{aligned}$$

Combining the boundary terms, eliminating κ_B with the definition of K_B, and re-arranging some terms yields the final expression of Hamilton's Law for an isotropic

beam:

$$\int_{t_1}^{t_2}\int_0^{\ell}[(\dot{\overline{\delta q}}_B^T-\overline{\delta q}_B^T\tilde{\Omega}_B-\overline{\delta\psi}_B^T\tilde{V}_B)P_B+(\dot{\overline{\delta\psi}}_B^T-\overline{\delta\psi}_B^T\tilde{\Omega}_B)H_B$$
$$-((\overline{\delta q}_B^T)'-\overline{\delta\psi}_B^T(\tilde{e}_1+\tilde{\gamma}_B)-\overline{\delta q}_B^T\tilde{K}_B)F_B-((\overline{\delta\psi}_B^T)'-\overline{\delta\psi}_B^T\tilde{K}_B)M_B$$
$$+\delta V_B^T(P_B-mV_B+m\tilde{\tilde{\xi}}_B\Omega_B)+\delta\Omega_B^T(H_B-I\Omega_B-m\tilde{\tilde{\xi}}_BV_B)$$
$$+\delta\gamma_B^T(A\gamma_B+B\kappa_B-F_B)+\delta K_B^T(B^T\gamma_B+D\kappa_B-M_B)$$
$$+\overline{\delta P}^T(v_b+\tilde{w}_bu_b-C^{bB}V_B)+\overline{\delta F}^T(C^{bB}(\gamma_B+e_1)-(e_1+\tilde{k}_bu_b))$$
$$+\overline{\delta H}^T(H^{Bb})^{-1}(C^{Bb}\omega_b-\Omega_B)+\overline{\delta M}^T(H^{Bb})^{-1}(K_B-C^{Bb}k_b)$$
$$-\dot{\overline{\delta P}}^Tu_b+(\overline{\delta F}^T)'u_b-\dot{\overline{\delta H}}^T\rho_{B/b}+(\overline{\delta M}^T)'\rho_{B/b}$$
$$-\delta U_G+\overline{\delta W}]\,dx_1dt$$
$$+\int_{t_1}^{t_2}(\overline{\delta q}_B^T\hat{F}_B+\overline{\delta\psi}_B^T\hat{M}_B-\overline{\delta F}^T\hat{u}_b-\overline{\delta M}^T\hat{\rho}_{B/b})|_0^{\ell}dt$$
$$+\int_0^{\ell}(\overline{\delta P}^T\hat{u}_b+\overline{\delta H}^T\hat{\rho}_{B/b}-\overline{\delta q}_B^T\hat{P}_B-\overline{\delta\psi}_B^T\hat{H}_B)|_{t_1}^{t_2}dx_1=0 \qquad (3.81)$$

3.13 Finite Element Discretization

Equation (3.81) represents the exact intrinsic equation for a beam. It is valid for beams with closed cross sections without any restrictions on warping. The form is similar to Hodges [26] with some minor differences including the choice of rotation parameters. In order to use this equation to obtain time accurate solutions to moving beam problems, a space-time finite element procedure is used. Integration is performed with shape functions of the lowest possible order. In this case, bilinear shape functions are used for the variational quantities $\overline{\delta q}_B$ and $\overline{\delta\psi}_B$, and linear shape functions are used for $\overline{\delta P}$, $\overline{\delta H}$, δV_B, $\delta\Omega_B$, $\overline{\delta F}$, $\overline{\delta M}$, $\delta\gamma_B$, and δK_B. For all non-variational quantities, constant shape functions are used. Define the non-dimensional time step

τ and non-dimensional length ε:

$$\begin{aligned}
\tau &= \frac{t - t_1}{t_2 - t_1} = \frac{t - t_1}{\Delta t} \\
\varepsilon &= \frac{x - x_1}{x_2 - x_1} = \frac{x - x_1}{\Delta x} \\
\dot{\tau} &= \frac{1}{\Delta t} \\
\varepsilon' &= \frac{1}{\Delta x}
\end{aligned} \tag{3.82}$$

Here the Δx refers to a distance along the reference line x_1, but the subscript is dropped to avoid confusion with x_1 and x_2, which are the spatial edges of an element.

The following are the shape functions for the variational quantities:

$$
\begin{aligned}
\overline{\delta q}_B &= \overline{\delta q}_1(1-\varepsilon)(1-\tau) + \overline{\delta q}_2\varepsilon(1-\tau) + \overline{\delta q}_3\varepsilon\tau + \overline{\delta q}_4(1-\varepsilon)\tau \\
\dot{\overline{\delta q}}_B &= \frac{1}{\Delta t}\left[\overline{\delta q}_1(\varepsilon-1) - \overline{\delta q}_2\varepsilon + \overline{\delta q}_3\varepsilon + \overline{\delta q}_4(1-\varepsilon)\right] \\
(\overline{\delta q}_B)' &= \frac{1}{\Delta x}\left[\overline{\delta q}_1(\tau-1) + \overline{\delta q}_2(1-\tau) + \overline{\delta q}_3\tau - \overline{\delta q}_4\tau\right] \\
\overline{\delta\psi}_B &= \overline{\delta\psi}_1(1-\varepsilon)(1-\tau) + \overline{\delta\psi}_2\varepsilon(1-\tau) + \overline{\delta\psi}_3\varepsilon\tau + \overline{\delta\psi}_4(1-\varepsilon)\tau \\
\dot{\overline{\delta\psi}}_B &= \frac{1}{\Delta t}(\overline{\delta\psi}_1(\varepsilon-1) - \overline{\delta\psi}_2\varepsilon + \overline{\delta\psi}_3\varepsilon + \overline{\delta\psi}_4(1-\varepsilon)) \\
(\overline{\delta\psi}_B)' &= \frac{1}{\Delta x}\left[\overline{\delta\psi}_1(\tau-1) + \overline{\delta\psi}_2(1-\tau) + \overline{\delta\psi}_3\tau - \overline{\delta\psi}_4\tau\right] \\
\delta V_B &= \delta V_1(1-\tau) + \delta V_2\tau \\
\delta\Omega_B &= \delta\Omega_1(1-\tau) + \delta\Omega_2\tau \\
\overline{\delta P} &= \overline{\delta P}_1(1-\tau) + \overline{\delta P}_2\tau \\
\overline{\delta H} &= \overline{\delta H}_1(1-\tau) + \overline{\delta H}_2\tau \\
\dot{\overline{\delta P}} &= \frac{1}{\Delta t}(\overline{\delta P}_2 - \overline{\delta P}_1) \\
\dot{\overline{\delta H}} &= \frac{1}{\Delta t}(\overline{\delta H}_2 - \overline{\delta H}_1) \\
\delta\gamma_B &= \delta\gamma_1(1-\varepsilon) + \delta\gamma_2\varepsilon \\
\delta K_B &= \delta K_1(1-\varepsilon) + \delta K_2\varepsilon \\
\overline{\delta F} &= \overline{\delta F}_1(1-\varepsilon) + \overline{\delta F}_2\varepsilon \\
\overline{\delta M} &= \overline{\delta M}_1(1-\varepsilon) + \overline{\delta M}_2\varepsilon \\
(\overline{\delta F})' &= \frac{1}{\Delta x}(\overline{\delta F}_2 - \overline{\delta F}_1) \\
(\overline{\delta M})' &= \frac{1}{\Delta x}(\overline{\delta M}_2 - \overline{\delta M}_1)
\end{aligned}
\tag{3.83}
$$

The following piecewise constant shape functions are defined for the generalized speeds and strains:

$$\begin{aligned} V_B &= \bar{V}_B & \gamma_B &= \bar{\gamma}_B \\ \Omega_B &= \bar{\Omega}_B & K_B &= \bar{K}_B \end{aligned} \tag{3.84}$$

Those quantities with boundary conditions are defined as follows:

$$P_B = \begin{cases} \hat{P}_1, & \tau = 0 \\ \bar{P}_B, & 0 < \tau < 1 \\ \hat{P}_2, & \tau = 1 \end{cases} \qquad F_B = \begin{cases} \hat{F}_1, & \varepsilon = 0 \\ \bar{F}_B, & 0 < \varepsilon < 1 \\ \hat{F}_2, & \varepsilon = 1 \end{cases}$$

$$H_B = \begin{cases} \hat{H}_1, & \tau = 0 \\ \bar{H}_B, & 0 < \tau < 1 \\ \hat{H}_2, & \tau = 1 \end{cases} \qquad M_B = \begin{cases} \hat{M}_1, & \varepsilon = 0 \\ \bar{M}_B, & 0 < \varepsilon < 1 \\ \hat{M}_2, & \varepsilon = 1 \end{cases}$$

$$u_b = \begin{cases} \hat{u}_1, & \tau = 0 \\ \hat{u}_3, & \tau = 1 \\ \hat{u}_4, & \varepsilon = 0 \\ \hat{u}_2, & \varepsilon = 1 \\ \bar{u}_b, & \text{otherwise} \end{cases} \qquad \rho_{B/b} = \begin{cases} \hat{\rho}_1, & \tau = 0 \\ \hat{\rho}_3, & \tau = 1 \\ \hat{\rho}_4, & \varepsilon = 0 \\ \hat{\rho}_2, & \varepsilon = 1 \\ \bar{\rho}_{B/b}, & \text{otherwise} \end{cases} \tag{3.85}$$

Figure 3.2 depicts a finite element with field quantities defined at space and time boundaries and within the element. P_B and H_B have discrete boundary conditions in the time domain, but are piecewise constant throughout a single beam element. They are not continuous between elements in space. The quantities V_B and Ω_B have no boundary conditions and are constant throughout the element. They are not continuous in space or time. Similarly, the quantities γ_B and K_B have no boundary conditions while F_B and M_B have discrete boundary conditions in space at the edges of

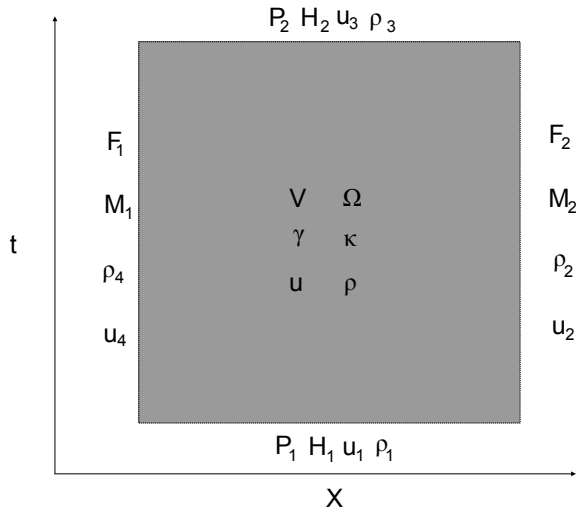


Figure 3.2: Finite element depiction: field quantities are either defined as constants within the element, or as spatial and temporal boundary conditions.

the element but are otherwise constant. Thus the strains are not continuous in space or time, and the forces and moments are discontinuous in time. The deformation field which consists of u_b and $\rho_{B/b}$ are piecewise constant through the element but have discrete boundary values at both space and time boundaries. They are piecewise continuous in space and time through their boundary conditions.

The advantage of using the weak form can now be seen, as all integration is performed by inspection. The following integrals are used:

$$\begin{aligned} \int_{t_1}^{t_2} dt &= \Delta t & \int_0^{\ell} dx_1 &= \Delta x \\ \int_{t_1}^{t_2} \tau dt &= \frac{1}{2}\Delta t & \int_0^{\ell} \varepsilon dx_1 &= \frac{1}{2}\Delta x \\ \int_{t_1}^{t_2} (1-\tau) dt &= \frac{1}{2}\Delta t & \int_0^{\ell} (1-\varepsilon) dx_1 &= \frac{1}{2}\Delta x \end{aligned} \tag{3.86}$$

Substitute the shape functions into equation (3.81), and expand the gravity and virtual work terms to obtain the following expression:

$$
\begin{aligned}
\int_{t_1}^{t_2}\int_0^{\ell} [(&\frac{1}{\Delta t}\left[\overline{\delta q}_1^T(\varepsilon-1)-\overline{\delta q}_2^T\varepsilon+\overline{\delta q}_3^T\varepsilon+\overline{\delta q}_4^T(1-\varepsilon)\right]\\
&-\left[\overline{\delta q}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta q}_2^T\varepsilon(1-\tau)+\overline{\delta q}_3^T\varepsilon\tau+\overline{\delta q}_4^T(1-\varepsilon)\tau\right]\tilde{\bar{\Omega}}\\
&-\left[\overline{\delta\psi}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta\psi}_2^T\varepsilon(1-\tau)+\overline{\delta\psi}_3^T\varepsilon\tau+\overline{\delta\psi}_4^T(1-\varepsilon)\tau\right]\tilde{\bar{V}})\bar{P}\\
&+(\frac{1}{\Delta t}\left[\overline{\delta\psi}_1^T(\varepsilon-1)-\overline{\delta\psi}_2^T\varepsilon+\overline{\delta\psi}_3^T\varepsilon+\overline{\delta\psi}_4^T(1-\varepsilon)\right]\\
&-\left[\overline{\delta\psi}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta\psi}_2^T\varepsilon(1-\tau)+\overline{\delta\psi}_3^T\varepsilon\tau+\overline{\delta\psi}_4^T(1-\varepsilon)\tau\right]\tilde{\bar{\Omega}})\bar{H}\\
&-(\frac{1}{\Delta x}\left[\overline{\delta q}_1^T(\tau-1)+\overline{\delta q}_2^T(1-\tau)+\overline{\delta q}_3^T\tau-\overline{\delta q}_4^T\tau\right]\\
&-\left[\overline{\delta\psi}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta\psi}_2^T\varepsilon(1-\tau)+\overline{\delta\psi}_3^T\varepsilon\tau+\overline{\delta\psi}_4^T(1-\varepsilon)\tau\right](\tilde{e}_1+\tilde{\bar{\gamma}}_B)\\
&-\left[\overline{\delta q}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta q}_2^T\varepsilon(1-\tau)+\overline{\delta q}_3^T\varepsilon\tau+\overline{\delta q}_4^T(1-\varepsilon)\tau\right]\tilde{\bar{K}})\bar{F}\\
&-(\frac{1}{\Delta x}\left[\overline{\delta\psi}_1^T(\tau-1)+\overline{\delta\psi}_2^T(1-\tau)+\overline{\delta\psi}_3^T\tau-\overline{\delta\psi}_4^T\tau\right]\\
&-\left[\overline{\delta\psi}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta\psi}_2^T\varepsilon(1-\tau)+\overline{\delta\psi}_3^T\varepsilon\tau+\overline{\delta\psi}_4^T(1-\varepsilon)\tau\right]\tilde{\bar{K}})\bar{M}\\
&+(\delta V_1^T(1-\tau)+\delta V_2^T\tau)(\bar{P}_B-m\bar{V}_B+m\tilde{\bar{\xi}}_B\bar{\Omega}_B)\\
&+(\delta\Omega_1^T(1-\tau)+\delta\Omega_2^T\tau)(\bar{H}_B-I\bar{\Omega}_B-m\tilde{\bar{\xi}}_B\bar{V}_B)\\
&+(\delta\gamma_1^T(1-\varepsilon)+\delta\gamma_2^T\varepsilon)(A\bar{\gamma}_B+B(\bar{K}_B-k_b)-\bar{F}_B)\\
&+(\delta K_1^T(1-\varepsilon)+\delta K_2^T\varepsilon)(B^T\bar{\gamma}_B+D(\bar{K}_B-k_b)-\bar{M}_B)\\
&+(\overline{\delta P}_1^T(1-\tau)+\overline{\delta P}_2^T\tau)(v_b+\tilde{w}_b\bar{u}_b-C^{bB}\bar{V}_B)\\
&+(\overline{\delta F}_1^T(1-\varepsilon)+\overline{\delta F}_2^T\varepsilon)(C^{bB}(\bar{\gamma}_B+e_1)-(e_1+\tilde{k}_b\bar{u}_b))\\
&+(\overline{\delta H}_1^T(1-\tau)+\overline{\delta H}_2^T\tau)(H^{Bb})^{-1}(C^{Bb}\omega_b-\bar{\Omega}_B)\\
&+(\overline{\delta M}_1^T(1-\varepsilon)+\overline{\delta M}_2^T\varepsilon)(H^{Bb})^{-1}(\bar{K}_B-C^{Bb}k_b)\\
&-\frac{1}{\Delta t}(\overline{\delta P}_2^T-\overline{\delta P}_1^T)\bar{u}_b+\frac{1}{\Delta x}(\overline{\delta F}_2^T-\overline{\delta F}_1^T)\bar{u}_b\\
&-\frac{1}{\Delta t}(\overline{\delta H}_2^T-\overline{\delta H}_1^T)\bar{\rho}_{B/b}+\frac{1}{\Delta x}(\overline{\delta M}_2^T-\overline{\delta M}_1^T)\bar{\rho}_{B/b}\\
&+\left[\overline{\delta q}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta q}_2^T\varepsilon(1-\tau)+\overline{\delta q}_3^T\varepsilon\tau+\overline{\delta q}_4^T(1-\varepsilon)\tau\right](f_B+mg_B)\\
&+\left[\overline{\delta\psi}_1^T(1-\varepsilon)(1-\tau)+\overline{\delta\psi}_2^T\varepsilon(1-\tau)+\overline{\delta\psi}_3^T\varepsilon\tau+\overline{\delta\psi}_4^T(1-\varepsilon)\tau\right](m_B\ldots
\end{aligned}
$$

$$
\begin{aligned}
&+ \tilde{\tilde{\xi}}_B m g_B)]\, dx_1 dt \\
&+ \int_{t_1}^{t_2} [(\overline{\delta q_1}^T (1-\varepsilon)(1-\tau) + \overline{\delta q_2}^T \varepsilon(1-\tau) + \overline{\delta q_3}^T \varepsilon\tau + \overline{\delta q_4}^T (1-\varepsilon)\tau)\hat{F}_B \\
&+ (\overline{\delta \psi_1}^T (1-\varepsilon)(1-\tau) + \overline{\delta \psi_2}^T \varepsilon(1-\tau) + \overline{\delta \psi_3}^T \varepsilon\tau + \overline{\delta \psi_4}^T (1-\varepsilon)\tau)\hat{M}_B \\
&- (\overline{\delta F_1}^T (1-\varepsilon) + \overline{\delta F_2}^T \varepsilon)\hat{u}_b - (\overline{\delta M_1}^T (1-\varepsilon) + \overline{\delta M_2}^T \varepsilon)\hat{\rho}_{B/b}]|_0^\ell dt \\
&+ \int_0^\ell ((\overline{\delta P_1}^T (1-\tau) + \overline{\delta P_2}^T \tau)\hat{u}_b + (\overline{\delta H_1}^T (1-\tau) + \overline{\delta H_2}^T \tau)\hat{\rho}_{B/b} \\
&- \left[\overline{\delta q_1}^T (1-\varepsilon)(1-\tau) + \overline{\delta q_2}^T \varepsilon(1-\tau) + \overline{\delta q_3}^T \varepsilon\tau + \overline{\delta q_4}^T (1-\varepsilon)\tau\right] \hat{P}_B \\
&- \left[\overline{\delta \psi_1}^T (1-\varepsilon)(1-\tau) + \overline{\delta \psi_2}^T \varepsilon(1-\tau) + \overline{\delta \psi_3}^T \varepsilon\tau + \overline{\delta \psi_4}^T (1-\varepsilon)\tau\right] \hat{H}_B)|_{t_1}^{t_2} dx_1 = 0 \quad (3.87)
\end{aligned}
$$

From here, the integrals are evaluated by inspection to get the final beam equation, which is grouped by variational quantities:

$$
\begin{aligned}
&\overline{\delta q_1}^T[\left(-\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{P}_B-\left(-\frac{\Delta t}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{F}_B\\
&+\frac{\Delta x}{2}\hat{P}_1-\frac{\Delta t}{2}\hat{F}_1+\frac{\Delta x\Delta t}{4}(f_B+mg_B)]\\
&+\overline{\delta q_2}^T[\left(-\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{P}_B-\left(\frac{\Delta t}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{F}_B\\
&+\frac{\Delta x}{2}\hat{P}_1+\frac{\Delta t}{2}\hat{F}_2+\frac{\Delta x\Delta t}{4}(f_B+mg_B)]\\
&+\overline{\delta q_3}^T[\left(\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{P}_B-\left(\frac{\Delta t}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{F}_B\\
&-\frac{\Delta x}{2}\hat{P}_2+\frac{\Delta t}{2}\hat{F}_2+\frac{\Delta x\Delta t}{4}(f_B+mg_B)]\\
&+\overline{\delta q_4}^T[\left(\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{P}_B-\left(-\frac{\Delta t}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{F}_B\\
&-\frac{\Delta x}{2}\hat{P}_2-\frac{\Delta t}{2}\hat{F}_1+\frac{\Delta x\Delta t}{4}(f_B+mg_B)]\\
&+\overline{\delta\psi_1}^T[-\frac{\Delta x\Delta t}{4}\tilde{\bar{V}}_B\bar{P}_B+\left(-\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{H}_B+\frac{\Delta x\Delta t}{4}(\tilde{e_1}+\tilde{\bar{\gamma}}_B)\bar{F}_B\\
&+\left(\frac{\Delta t}{2}+\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{M}_B+\frac{\Delta x}{2}\hat{H}_1-\frac{\Delta t}{2}\hat{M}_1+\frac{\Delta x\Delta t}{4}(m_B+\tilde{\bar{\xi}}_Bmg_B)]\\
&+\overline{\delta\psi_2}^T[-\frac{\Delta x\Delta t}{4}\tilde{\bar{V}}_B\bar{P}_B+\left(-\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{H}_B+\frac{\Delta x\Delta t}{4}(\tilde{e_1}+\tilde{\bar{\gamma}}_B)\bar{F}_B\\
&-\left(\frac{\Delta t}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{M}_B+\frac{\Delta x}{2}\hat{H}_1+\frac{\Delta t}{2}\hat{M}_2+\frac{\Delta x\Delta t}{4}(m_B+\tilde{\bar{\xi}}_Bmg_B)]\\
&+\overline{\delta\psi_3}^T[-\frac{\Delta x\Delta t}{4}\tilde{\bar{V}}_B\bar{P}_B+\left(\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{H}_B+\frac{\Delta x\Delta t}{4}(\tilde{e_1}+\tilde{\bar{\gamma}}_B)\bar{F}_B\\
&-\left(\frac{\Delta t}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{M}_B-\frac{\Delta x}{2}\hat{H}_2+\frac{\Delta t}{2}\hat{M}_2+\frac{\Delta x\Delta t}{4}(m_B+\tilde{\bar{\xi}}_Bmg_B)]\\
&+\overline{\delta\psi_4}^T[-\frac{\Delta x\Delta t}{4}\tilde{\bar{V}}_B\bar{P}_B+\left(\frac{\Delta x}{2}-\frac{\Delta x\Delta t}{4}\tilde{\bar{\Omega}}_B\right)\bar{H}_B+\frac{\Delta x\Delta t}{4}(\tilde{e_1}+\tilde{\bar{\gamma}}_B)\bar{F}_B\\
&+\left(\frac{\Delta t}{2}+\frac{\Delta x\Delta t}{4}\tilde{\bar{K}}_B\right)\bar{M}_B-\frac{\Delta x}{2}\hat{H}_2-\frac{\Delta t}{2}\hat{M}_1+\frac{\Delta x\Delta t}{4}(m_B+\tilde{\bar{\xi}}_Bmg_B)]\\
&+(\overline{\delta V_1}^T+\overline{\delta V_2}^T)[\frac{\Delta x\Delta t}{2}(\bar{P}_B-m\bar{V}_B+m\tilde{\bar{\xi}}_B\bar{\Omega}_B)]\\
&+(\overline{\delta\Omega_1}^T+\overline{\delta\Omega_2}^T)[\frac{\Delta x\Delta t}{2}(\bar{H}_B-I\bar{\Omega}_B-m\tilde{\bar{\xi}}_B\bar{V}_B)]\\
&+(\overline{\delta\gamma_1}^T+\overline{\delta\gamma_2}^T)[\frac{\Delta x\Delta t}{2}(A\bar{\gamma}_B+B(\bar{K}_B-k_b)-\bar{F}_B)]\dots
\end{aligned}
$$

$$
\begin{aligned}
&+ (\overline{\delta K}_1^T + \overline{\delta K}_2^T)[\frac{\Delta x \Delta t}{2}(B^T \bar{\gamma}_B + D(\bar{K}_B - k_b) - \bar{M}_B)] \\
&+ \overline{\delta P}_1^T[\frac{\Delta x \Delta t}{2}(v_b + \tilde{\omega}_b \bar{u}_b - C^{bB}\bar{V}_B) + \Delta x(\bar{u}_b - \hat{u}_1)] \\
&+ \overline{\delta P}_2^T[\frac{\Delta x \Delta t}{2}(v_b + \tilde{\omega}_b \bar{u}_b - C^{bB}\bar{V}_B) + \Delta x(\hat{u}_3 - \bar{u}_b)] \\
&+ \overline{\delta H}_1^T[\frac{\Delta x \Delta t}{2}(H^{Bb})^{-1}(C^{Bb}\omega_b - \bar{\Omega}_B) + \Delta x(\bar{\rho}_{B/b} - \hat{\rho}_1)] \\
&+ \overline{\delta H}_2^T[\frac{\Delta x \Delta t}{2}(H^{Bb})^{-1}(C^{Bb}\omega_b - \bar{\Omega}_B) + \Delta x(\hat{\rho}_3 - \bar{\rho}_{B/b})] \\
&+ \overline{\delta F}_1^T[\frac{\Delta x \Delta t}{2}(C^{bB}(\bar{\gamma}_B + e_1) - (e_1 + \tilde{k}_b \bar{u}_b)) + \Delta t(\hat{u}_4 - \bar{u}_b)] \\
&+ \overline{\delta F}_2^T[\frac{\Delta x \Delta t}{2}(C^{bB}(\bar{\gamma}_B + e_1) - (e_1 + \tilde{k}_b \bar{u}_b)) + \Delta t(\bar{u}_b - \hat{u}_2)] \\
&+ \overline{\delta M}_1^T[\frac{\Delta x \Delta t}{2}(H^{Bb})^{-1}(\bar{K}_B - C^{Bb}k_b) + \Delta t(\hat{\rho}_4 - \bar{\rho}_{B/b})] \\
&+ \overline{\delta M}_2^T[\frac{\Delta x \Delta t}{2}(H^{Bb})^{-1}(\bar{K}_B - C^{Bb}k_b) + \Delta t(\bar{\rho}_{B/b} - \hat{\rho}_2)] = 0
\end{aligned} \tag{3.88}
$$

In this equation it is clear that all expressions with separate coefficients must individually be equal to zero. This generates a system of 20 equations. However, only 18 of these equations are independent. The four equations generated from the $\overline{\delta q}^T$ coefficients are redundant, as are the equations from $\overline{\delta \psi}^T$. Four equations and four unknowns are eliminated from this system by subtracting combinations of the $\overline{\delta q}_B$ and $\overline{\delta \psi}_B$ terms to obtain the following relationships:

$$
\begin{aligned}
\bar{F}_B &= \frac{1}{2}(\hat{F}_1 + \hat{F}_2) \\
\bar{M}_B &= \frac{1}{2}(\hat{M}_1 + \hat{M}_2) \\
\bar{P}_B &= \frac{1}{2}(\hat{P}_1 + \hat{P}_2) \\
\bar{H}_B &= \frac{1}{2}(\hat{H}_1 + \hat{H}_2)
\end{aligned} \tag{3.89}
$$

The following is also determined by adding combinations of the last eight terms of (3.88):

$$\bar{u}_b = \frac{1}{4}(\hat{u}_1 + \hat{u}_2 + \hat{u}_3 + \hat{u}_4)$$

$$\bar{\rho}_b = \frac{1}{4}(\hat{\rho}_1 + \hat{\rho}_2 + \hat{\rho}_3 + \hat{\rho}_4) \tag{3.90}$$

$$\tag{3.91}$$

These two relationships will not be used to eliminate equations, since the deformation variables cannot be completely eliminated from the final equation set. Eliminating the redundant equations and $\bar{F}_B$, $\bar{M}_B$, $\bar{P}_B$, and $\bar{H}_B$ yields the final set of 14 equations

and 14 unknowns for each element:

$$\frac{\Delta x}{4}(\hat{P}_1 - \hat{P}_2) - \frac{\Delta x \Delta t}{8}[\tilde{\bar{\Omega}}_B(\hat{P}_1 + \hat{P}_2) - \tilde{\bar{K}}_B(\hat{F}_1 + \hat{F}_2)] + \frac{\Delta t}{4}(\hat{F}_2 - \hat{F}_1)$$
$$+ \frac{\Delta x \Delta t}{4}(f_B + m g_B) = 0 \tag{3.92}$$

$$\frac{\Delta x \Delta t}{8}[-\tilde{\bar{V}}_B(\hat{P}_1 + \hat{P}_2) - \tilde{\bar{\Omega}}_B(\hat{H}_1 + \hat{H}_2) + (\tilde{e_1} + \tilde{\bar{\gamma}}_B)(\hat{F}_1 + \hat{F}_2) + \tilde{\bar{K}}_B(\hat{M}_1 + \hat{M}_2)]$$
$$+ \frac{\Delta x}{4}(\hat{H}_1 - \hat{H}_2) + \frac{\Delta t}{4}(\hat{M}_2 - \hat{M}_1) + \frac{\Delta x \Delta t}{4}(m_B + \tilde{\bar{\xi}}_B m g_B) = 0 \tag{3.93}$$

$$\frac{1}{2}(\hat{P}_1 + \hat{P}_2) - m\bar{V}_B + m\tilde{\bar{\xi}}_B\bar{\Omega}_B = 0 \tag{3.94}$$

$$\frac{1}{2}(\hat{H}_1 + \hat{H}_2) - I\bar{\Omega}_B - m\tilde{\bar{\xi}}_B\bar{V}_B = 0 \tag{3.95}$$

$$A\bar{\gamma}_B + B(\bar{K}_B - k_b) - \frac{1}{2}(\hat{F}_1 + \hat{F}_2) = 0 \tag{3.96}$$

$$B^T\bar{\gamma}_B + D(\bar{K}_B - k_b) - \frac{1}{2}(\hat{M}_1 + \hat{M}_2) = 0 \tag{3.97}$$

$$\frac{\Delta t}{2}(v_b + \tilde{\omega}_b\bar{u}_b - C^{bB}\bar{V}_B) + \bar{u}_b - \hat{u}_1 = 0 \tag{3.98}$$

$$\frac{\Delta t}{2}(v_b + \tilde{\omega}_b\bar{u}_b - C^{bB}\bar{V}_B) + \hat{u}_3 - \bar{u}_b = 0 \tag{3.99}$$

$$\frac{\Delta t}{2}(H^{Bb})^{-1}(C^{Bb}\omega_b - \bar{\Omega}_B) + \bar{\rho}_{B/b} - \hat{\rho}_1 = 0 \tag{3.100}$$

$$\frac{\Delta t}{2}(H^{Bb})^{-1}(C^{Bb}\omega_b - \bar{\Omega}_B) + \hat{\rho}_3 - \bar{\rho}_{B/b} = 0 \tag{3.101}$$

$$\frac{\Delta x}{2}(C^{bB}(\bar{\gamma}_B + e_1) - (e_1 + \tilde{k}_b\bar{u}_b)) + \hat{u}_4 - \bar{u}_b = 0 \tag{3.102}$$

$$\frac{\Delta x}{2}(C^{bB}(\bar{\gamma}_B + e_1) - (e_1 + \tilde{k}_b\bar{u}_b)) + \bar{u}_b - \hat{u}_2 = 0 \tag{3.103}$$

$$\frac{\Delta x}{2}(H^{Bb})^{-1}(\bar{K}_B - C^{Bb}k_b) + \hat{\rho}_4 - \bar{\rho}_{B/b} = 0 \tag{3.104}$$

$$\frac{\Delta x}{2}(H^{Bb})^{-1}(\bar{K}_B - C^{Bb}k_b) + \bar{\rho}_{B/b} - \hat{\rho}_2 = 0 \tag{3.105}$$

Note that each unknown is a vector with three components, and thus there are actually 42 variables that must be solved in each element. The first two equations account for externally applied loads and relate forces, momenta, velocities, and strains. Equations (3.94 - 3.97) are the constitutive equations that relate velocities to momenta and internal forces to strains. Equations (3.98 - 3.101) are kinematic equations that relate velocities to displacements, and (3.102 - 3.105) are strain-displacement relationships.

Equations (3.92 - 3.105) contain 22 separate vector quantities, some within the element and some prescribed on the boundaries. There are significant differences between this formulation and a more traditional displacement formulation. Here forces, momenta, strains, velocities, and displacements are all propagated as separate field quantities. While adding complexity, it also allows natural boundary conditions to be enforced explicitly in the equations. Note also that the values are not defined at "nodes", as would be the case with traditional finite elements, but only along the edges or within the element. Displacements are defined on all four edges (two spatial edges and two temporal edges) and also in the center. The values at the edges are piecewise constant and the interpolation polynomial for displacement within the element is also a constant. However, defining the displacement at both the edges and in the center provides a bilinear interpolation function. In other words, while the value of displacement is assumed to be constant within the element and is the average of all the boundaries, one may use the boundary values to obtain a linear solution in both space and time. This greatly improves convergence for beam problems where the exact solution is of a higher order. This same concept applies to momenta, which are defined along time boundaries, and internal forces, which are defined at the spatial boundaries. It does not apply to generalized velocities or strains, which are only defined inside the element.

3.14 Equations for Steady-State Analysis

Given a set of prescribed conditions, the initial displacements and momenta may not be known. In this case, a complete set of initial conditions does not exist and must be solved prior to the dynamic analysis. This is accomplished by solving a steady-state problem, where equations (3.92 - 3.105) are appended with the following

constraints:

$$\begin{aligned}\hat{\rho}_1 &= \hat{\rho}_3 \\ \hat{u}_1 &= \hat{u}_3 \\ \hat{P}_1 &= \hat{P}_3 \\ \hat{H}_1 &= \hat{H}_3\end{aligned} \tag{3.106}$$

The relationships in (3.106) are supplied instead of initial conditions. This technique will also yield solutions to static problems. The result gives the initial conditions for the dynamic problem. Thus there are two different solvers used, one steady and one dynamic, and the steady solution is obtained first to supply the full set of initial conditions for the dynamic solver.

3.15 Solution Algorithm

Finite element discretization of Hamilton's Weak Principle yields a set of nonlinear algebraic equations. It is both an initial value problem in time and a boundary value problem in space. A typical approach for this type of problem is to solve the boundary problem for the initial time step, use the solution as the initial condition for the next step, and iterate in time [3, 23, 34] (see Figure 3.3). It requires that the full configuration of the beam is solved at each time interval.

The beam is discretized into N_x elements along its reference line and solved over a time interval of N_t increments. In this problem there are $22N_x$ field quantities and $14N_x$ equations. To fully define the problem there must also exist $4N_x$ initial conditions and $4N_t$ boundary conditions (a set of four boundary conditions during each time step). The initial conditions are $\hat{\rho}_1$, $\hat{u}_1$, $\hat{P}_1$, and $\hat{H}_1$ in each spatial element. Boundary conditions are specified as root or tip conditions, and come from $\hat{\rho}_2$, $\hat{\rho}_4$, $\hat{u}_2$, $\hat{\rho}_4$, $\hat{F}_1$, $\hat{F}_2$, $\hat{M}_1$, or $\hat{M}_2$, depending on the problem. For a cantilevered beam, the root displacements are fixed and the tip forces and moments are prescribed. This generates four boundary conditions. The aerospace application of immediate interest (rotor

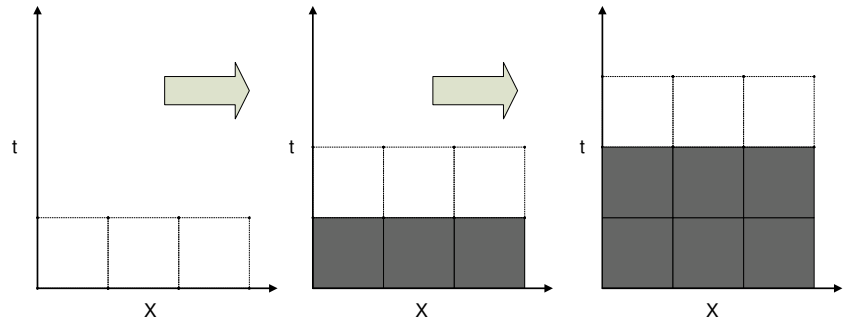


Figure 3.3: Time marching procedure: all elements within a single time step are solved simultaneously, and the solution provides the initial conditions for the next time step. The algorithm marches in time until the complete solution is determined.

blades and flexible wings) is a cantilevered problem, and thus it is used exclusively as the boundary condition for this research. However, other combinations would be used for different boundary conditions. The beam elements also share boundary values along their edges. For a continuous beam the values of q_b, $\rho_{B/b}$, F_B, and M_B are shared at element edges. For various types of joints (such as a hinge), some of these quantities will be shared and others may be prescribed. Figure 3.4 shows a beam element and its shared properties with the adjoining elements.

After applying the 4 boundary conditions, $4N_x$ initial conditions, and $4(N_x - 1)$ shared edge conditions, only $14N_x$ unknowns remain. The algorithm must therefore solve a system of $14N_x$ equations and $14N_x$ unknowns at each time step. This is accomplished by posing the system of equations as an unconstrained minimization problem which may be solved numerically. The problem is of the form,

$$F(x) = 0$$

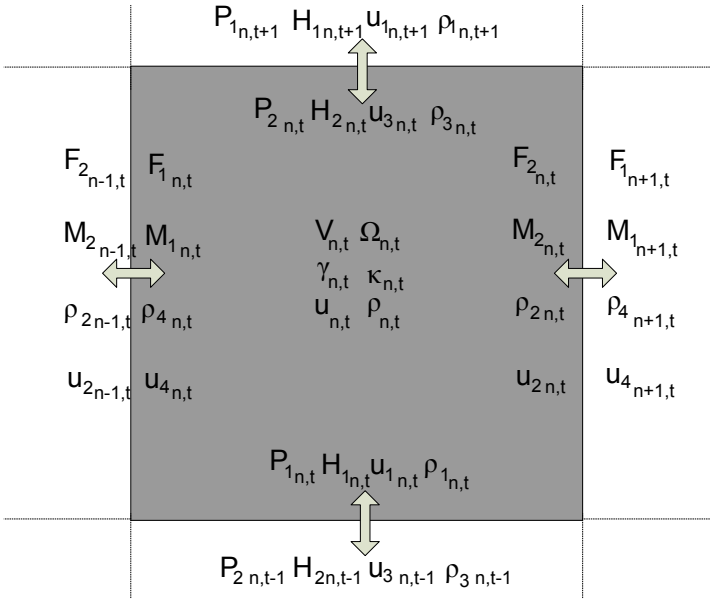


Figure 3.4: Each element shares properties along its edges (indicated by arrows) with the adjoining elements.

where $F(x)$ is a vector containing the residuals of equations (3.92) through (3.105), which must be minimized. There are many algorithms for solving this problem. In this research it is solved with a Gauss-Newton method that is implemented in MATLAB®. While the details of the numerical algorithm are outside the scope of this research, the problem must be preconditioned so that the algorithm will converge. This requires a good initial guess for the solution and appropriate scaling in the state vector.

3.16 Scaling and Units

Unconstrained minimization algorithms perform poorly when the variables are inconsistent in magnitude, and often fail to find the solution [35]. Unfortunately, units of force are generally very large and units of displacement are very small. For dynamic problems the gradients are also a factor and very small units of time are needed to accurately capture information. Using typical English units of pounds, inches, and seconds, many of the test cases will fail to converge. However, when the basic units are changed so that the state vector is of a consistent magnitude, the algorithm will perform well. For dynamic problems this requires the use of milliseconds as the basic

unit of time. The unit of force changes from a $sl \cdot in \cdot sec^{-2}$, where a slinch ($sl$) is 1/12th of a slug, to a $sl \cdot in \cdot ms^{-2}$. This is equivalent to 1x10$^6 lb$, or $1Mlb$. However, in other problems (usually static or steady-state) it is more convenient to use pounds or kips and keep the units of time in seconds. One must know the magnitude of the solution ahead of time in order to choose the correct units. Otherwise, the problem may be poorly scaled and this could prevent the algorithm from converging. The test cases illustrate this.

3.17 Choosing the Initial Guess

In order to implement the algorithm, an initial guess is used to begin searching for the solution. Often the convergence of the problem depends on how close the initial guess is to the solution, and in some cases a poor initial guess will either prevent the algorithm from converging or cause it to converge on a minimum that is not the correct solution. For a typical static problem, the initial guess is set to zero. For some problems, however, this is not close enough. There are two techniques used to deal with this problem. One is used for static problems, the other for time accurate problems.

For highly nonlinear problems where the solution is nowhere near zero, the problem is solved incrementally. The process is similar to incremental loading which is also used in traditional finite element analysis programs for nonlinear problems. First the applied load is divided into increments, then solved at the smallest increment using an initial guess of zero. The solution is then used as the initial guess for the next increment. The algorithm continues in this manner until reaching the final loading condition, which yields the solution. Solving incrementally keeps the initial guess close to the solution.

For dynamic problems, the gradients in displacement and force can be very large. If the step size is small enough, the previous time step can be used as the initial guess for the solution. To improve this further, numerical derivatives are calculated using data from the previous three time steps. First and second derivatives are used to

project where the next guess should be by extrapolating the data. This places the guess closer to where the solution should be. It does not guarantee convergence, nor does it guarantee unconditional stability in the solution.

Let X_i represent the state vector of unknowns at time step i. If the solution to both X_i and X_{i-1} has been determined, then the first derivative $\dot{X}_i$ is used to estimate the guess for the next time step (X_{i+1}) as:

$$\begin{aligned} \dot{X}_i &= \frac{1}{\Delta t}(X_i - X_{i-1}) \\ X_{i+1_{guess}} &= X_i + \Delta t \dot{X}_i \\ X_{i+1_{guess}} &= 2X_i - X_{i-1} \end{aligned} \tag{3.107}$$

Likewise, if the previous two time steps are known, second derivative information is also available as $\ddot{X}_i$.

$$\begin{aligned} \ddot{X}_i &= \frac{1}{\Delta t}(\dot{X}_i - \dot{X}_{i-1}) \\ \ddot{X}_i &= \frac{1}{(\Delta t)^2}(X_i - 2X_{i-1} + X_{i-2}) \\ X_{i+1_{guess}} &= X_i + \Delta t(\dot{X}_i + \Delta t \ddot{X}_i) \\ X_{i+1_{guess}} &= 3X_i - 3X_{i-1} + X_{i-2} \end{aligned} \tag{3.108}$$

3.18 Analytical Jacobian

Numerical algorithms for solving systems of nonlinear algebraic equations calculate a Jacobian matrix, which contains the partial derivative of each equation with respect to each variable. This provides the search direction for most methods. In this case, the Jacobian is a $42N_x$ by $42N_x$ square matrix. While most algorithms can calculate this with finite difference techniques, using the analytical Jacobian greatly improves the speed of the algorithm. An analytical Jacobian is used in this research; however, due to its size it is not practical to show the development here. Appendix

D, which contains a listing of the computer program, shows the equations used to determine the Jacobian.

3.19 Error Tolerance Criteria

Because the system of equations is solved as an unconstrained minimization problem, it will continue to search for a solution until the residuals are sufficiently small. At this point the algorithm will determine that it has reached a minimum and stop. Because the quantities Δx and Δt appear in equations (3.92 - 3.105), the element size affects the magnitude of these residuals. As more elements are used, the decrease in Δx also decreases the residuals, and this has the undesired effect of increasing error tolerance. It will cause the algorithm to produce inaccurate solutions for large mesh sizes unless accounted for. This is accomplished by manually decreasing the error tolerance in the algorithm, or by making it a function of the element size. Either method is sufficient.

3.20 Conservation Laws

It is a common practice to apply conservation laws to finite element discretization schemes [28]. This allows the numerical preservation of energy to be demonstrated in the solution [11, 20]. These algorithms are sometimes referred to as energy preserving schemes. The current research does not prove the conservation of energy mathematically, but it is demonstrated for certain cases numerically. Bauchau [10,11] discusses the problems that occur with energy preserving schemes when trying to solve complex, flexible body systems with multiple elements. When energy preservation algorithms are used, high frequency modes are introduced into the system as the number of elements is increased. High frequency numerical dissipation is required to make the algorithm useful. Bauchau presents several energy decaying formulations which correct this problem. In general, they apply to displacement finite element formulations, involving a tuning parameter which determines the amount of dissipation

through each time interval. No energy decaying formulation has yet been developed for mixed finite element formulations.

IV. Analysis and Results

In this chapter the computational algorithm is applied to several classes of problems to evaluate its capabilities and robustness, and the results and convergence criteria are presented. The scenarios are given in order of increasing complexity, beginning with the simplest static equilibrium problems and finishing with more complex problems of dynamic motion.

4.1 Axial Rods

The initial tests of the algorithm were accomplished by obtaining static solutions to various problems. This verifies the ability to obtain complete initial conditions for a structural problem before proceeding with the time accurate simulation. For all static and steady-state problems, equations (3.92 - 3.105) are solved with (3.106) as a constraint. The basic units for axial rod problems are pounds, inches, and seconds. No convergence problems associated with scaling were encountered, although it was later determined that greater efficiency is attainable with units scaled such that the relative magnitudes of force, strain, and displacement are the same. Problems with tip loads are modeled by specifying the force as a boundary condition, while distributed loads are modeled with virtual work.

The simplest configuration to test is the deformation of a fixed-free rod, which reduces the system to $14N_x$ linear equations. First, a 12-inch aluminum rod with a circular cross section is loaded axially and in torsion. It is cantilevered on one end and the loads are applied at the tip. Figure 4.1a illustrates these conditions. The following material properties were used for aluminum: $\rho = 0.098\, lb_f/in^3$, $E = 10^7$ psi, and $\nu = 0.35$. For the first case, P is a 1000 lb load and T is a 1000 in-lb torque. Because this is a constant strain problem, the model converges to the exact solution using a single element (see Figure 4.2). Analytical solutions are obtained from the

(a) Tip loads

(b) Distributed loads

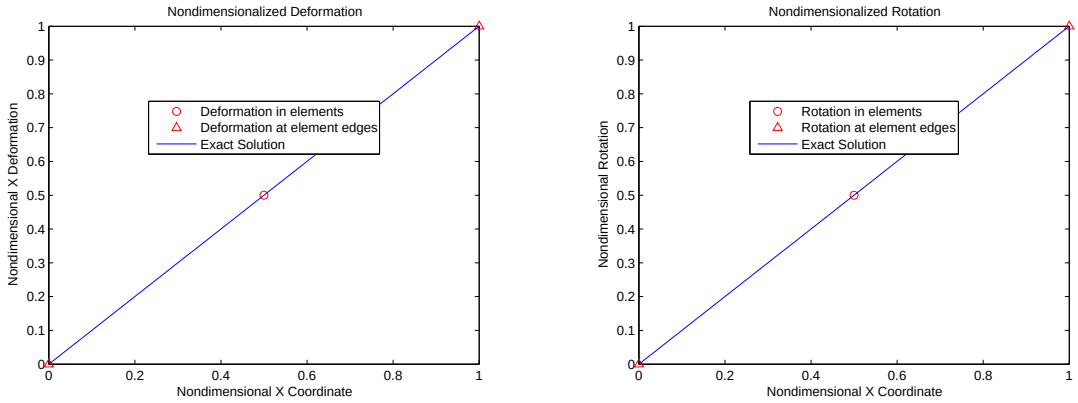
Figure 4.1: Axial rod geometry and loading configurations.

following equations, which can be found in most engineering mechanics textbooks [25]:

$$\begin{aligned} F_x(x) &= P & M_x(x) &= T \\ \gamma_x(x) &= \frac{P}{EA} & \kappa_x(x) &= \frac{T}{GJ} \\ u_x(x) &= \gamma_x x & \theta_x(x) &= \kappa_x x \end{aligned} \tag{4.1}$$

The quantity F_x is the internal force resultant, M_x is the internal torsion, κ_x and γ_x are the corresponding force and moment strains, and u_x and θ_x are the displacement and twist. Because forces, strains, and displacements are calculated as independent field quantities, they are solved for simultaneously. Complete solutions for each quantity are located in Appendix A. Each solution is non-dimensionalized by its maximum value for comparison. The problem was also solved using meshes of 5, 10, 20, and 30 elements to determine if any problems would develop as the number of elements increased. This led to the discovery that decreasing element size (Δx) can impact convergence, if the error tolerance in the numerical solver is not set low enough (as discussed in chapter 3).

In the second case, a distributed load of 83.33 lb/in and a distributed torque of 83.33 in-lb/in are applied according to Figure 4.1b. Distributed loads are applied using virtual work, and therefore it is not necessary to resolve them to equivalent



(a) Deformation (b) Twist

Figure 4.2: Nondimensionalized deformation and twist for a rod with an axial tip load and torque.

nodal loads, as is typical with conventional finite element methods. The values for distributed forces and moments in the equations are specified by the user. The displacements in this problem are quadratic, and the data converges to the exact solution as more elements are used in the mesh. Mean error (E) is calculated using the difference between the calculated and exact solutions. If k is the number of data points, $F'(x)$ is the calculated solution, $F(x)$ is the exact solution, and x_n is the x-coordinate along the length of the beam, then the mean error for any field quantity is obtained as follows:

$$E = \frac{1}{k}\sum_{n=1}^{k}\left(\frac{||F'(x_n) - F(x_n)||}{F(x_n)}\right) \tag{4.2}$$

This relationship is defined for all data points except when the exact solution is zero. Generally, when this occurs the numerical solution is also zero (or machine zero) and thus the error is negligible. Convergence is demonstrated using meshes ranging from

1 to 30 elements. The following analytical solutions are used for comparison [25]:

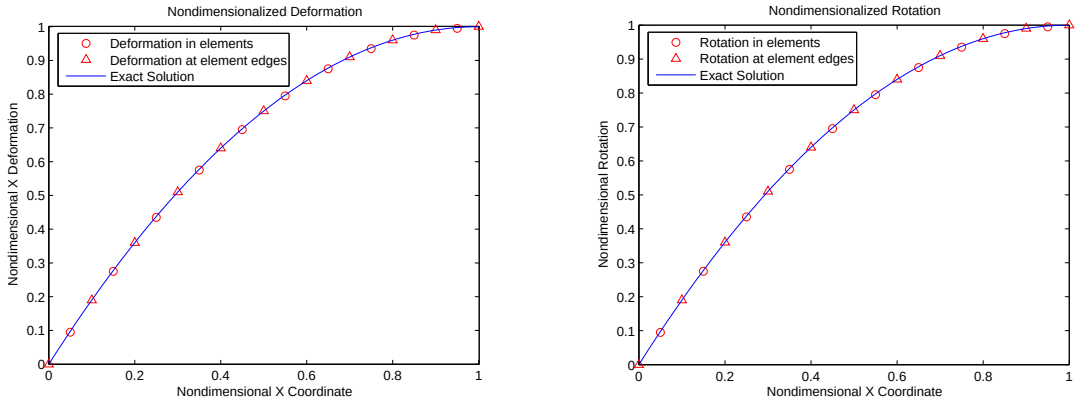
$$
\begin{aligned}
F_x(x) &= F_x(0) - \int_0^x p_x(x)\,dx & \qquad M_x(x) &= M_x(0) - \int_0^x t_x(x)\,dx \\
\gamma_x(x) &= \frac{F_x(x)}{EA} & \qquad \kappa_x(x) &= \frac{M_x(x)}{GJ} \\
u_x(x) &= \int_0^x \gamma_x(x)dx & \qquad \theta_x(x) &= \int_0^x \kappa_x(x)dx
\end{aligned}
\tag{4.3}
$$

Table 4.1 illustrates the convergence dynamics for all field variables. The purpose of the numerical solver is to find the root of the equations by minimizing the norm of the residuals within a specific tolerance. As more elements are used, more variables are introduced into the state vector. As more variables are used, the error in these values can increase without exceeding the termination criteria of the solver. This is aggravated by the fact that finer meshes have more points closer to the root, where the strains are much smaller. The percent error in these points is generally larger because the denominator of equation (4.2) is much smaller than at other points along the beam. The result is that certain field variables do not converge numerically. Instead, they have a tendency to increase, though the error itself is not significant. This appears to be an issue with the tolerance in the solver and not with the system equations. While this does not illustrate convergence in a pure mathematical sense, it is sufficient to show that the algorithm will converge to the correct solution within a specific tolerance.

As expected, the greatest error occurs in the displacements, where the exact solution is of the highest order (in this case a second order polynomial). Less than one percent error appears in the mesh with 10 elements, and these results are depicted in Figure 4.3. Note that the error at element edges is significantly smaller than the centroids. This is an advantage of enforcing boundary conditions at the edges. Because the internal displacement of an element is determined by the average displacement of the two edges, it will always have error even if the solution at the edges is exact (unless the exact solution is linear or constant). Because the greatest error occurs

Table 4.1: Mean Error in each Variable for an Axial Rod with Distributed Loads

Elements	$\bar{u}$	$\hat{u}$	$\bar{\rho}$	$\hat{\rho}$	$\bar{\gamma}$	$\bar{\kappa}$	$\bar{F}$	$\hat{F}$	$\bar{M}$	$\hat{M}$
1	0.3333	2E-5	0.3333	2E-5	3E-5	4E-6	0	0	0	0
5	0.0213	1E-5	0.0213	8E-6	1E-5	8E-6	2E-16	1E-16	2E-16	1E-16
10	0.0062	1E-5	0.0062	8E-6	1E-5	1E-5	1E-16	6E-17	1E-16	6E-17
20	0.0018	1E-5	0.0018	1E-5	1E-5	1E-5	1E-16	8E-17	1E-16	8E-17
30	0.0008	1E-5	0.0008	9E-6	9E-6	2E-5	6E-6	6E-6	6E-6	6E-6

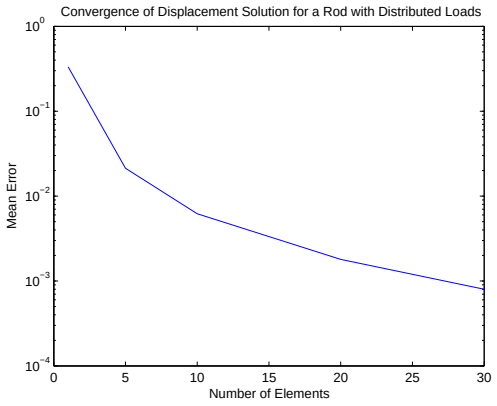


(a) Deformation (b) Twist

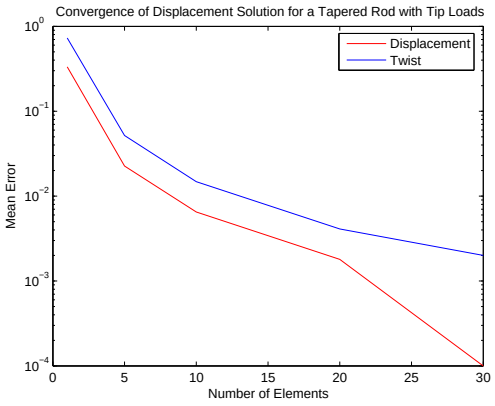
Figure 4.3: Nondimensionalized deformation and twist for a rod with a distributed axial load and torque.

in the displacement at the centroids, this quantity is used to determine convergence. Figure 4.4 illustrates the convergence in the displacement solutions for problems with rods. Since the error is nearly identical in both the displacement and twist for the uniform rod with distributed loads, only the displacement error is presented.

For the final case, a 24-inch tapered aluminum rod is subject to a 1000 lb tip load and 1000 in-lb torque as shown in Figure 4.5. Individual elements have a constant cross section, which is determined by the average diameter over the length they span. As more elements are used the taper is modeled with greater resolution, and the solution converges as shown in Table 4.2. The following relationships are used to obtain the analytical solution for a cantilevered rod with a circular cross section



(a) Uniform Rod, Distributed Loads



(b) Tapered Rod, Tip Loads

Figure 4.4: Mean error versus number of elements for problems with rods.

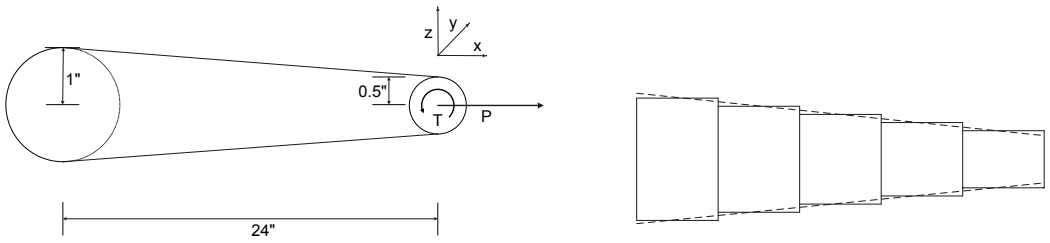
Table 4.2: Mean Error in each Variable for the Tapered Rod Problem

Elements	$\bar{u}$	$\hat{u}$	$\bar{\gamma}$	$\bar{\rho}$	$\hat{\rho}$	$\bar{\kappa}$
1	0.3333	0.1111	7E-6	0.7297	0.3227	2E-6
5	0.0226	0.0040	1E-5	0.0518	0.0139	6E-6
10	0.0065	0.0001	7E-6	0.0148	0.0034	9E-6
20	0.0018	2E-4	6E-6	0.0041	0.0008	1E-7
30	0.0001	1E-4	7E-6	0.0020	0.0004	8E-6

$A(x)$, radius $r(x)$, length L, torsional rigidity $J(x)$, and constant taper ratio t:

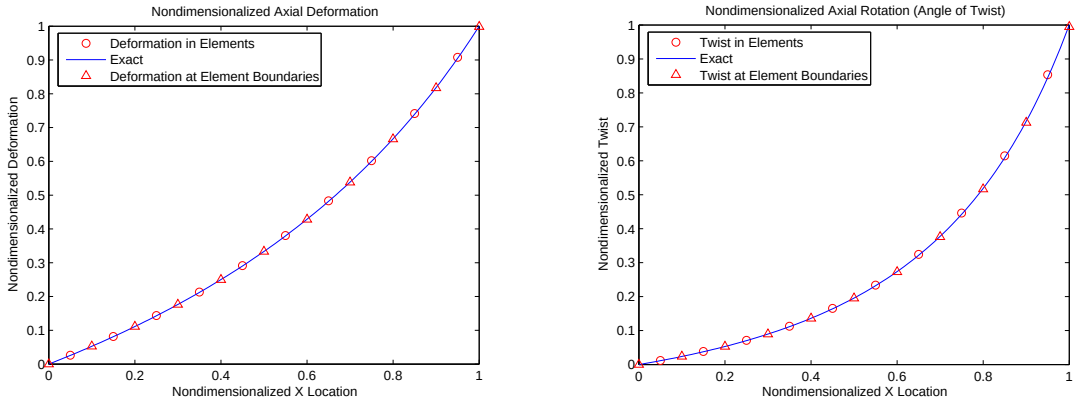
$$\begin{aligned} A(x) &= \pi \left[r_0 \left(1 - t\frac{x}{L} \right) \right]^2 & F(x) &= P \\ J(x) &= \frac{\pi}{2} \left[r_0 \left(1 - t\frac{x}{L} \right) \right]^4 & M(x) &= T \\ \gamma(x) &= \frac{P}{EA(x)} & u(x) &= \int_0^x \gamma(x)\, dx \\ \kappa(x) &= \frac{T}{GJ(x)} & \theta(x) &= \int_0^x \kappa(x)\, dx \end{aligned} \tag{4.4}$$

The convergence for this solution is based on the mean error from the exact solution, where the displacements at the element centroids have the greatest error. As in the previous problem, the displacement solution is converged (within one percent error) at 10 elements (see Figure 4.6). Since the torsional rigidity is a fourth order



(a) Exact Geometry (b) Finite Element Representation

Figure 4.5: Geometry of a tapered aluminum rod with a 50% taper ratio and the finite element representation with 5 elements



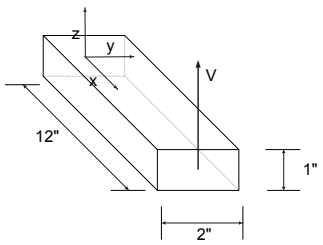
(a) Deformation (b) Twist

Figure 4.6: Nondimensionalized deformation of a tapered rod with an axial tip load and torque.

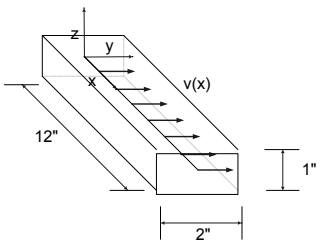
function, the angle of twist does not converge as quickly. It is within a percent at 20 elements. The complete solution is located in Appendix A.

4.2 Timoshenko Beams

Next, the algorithm is applied to problems of transverse loading and bending. These problems are nonlinear and have more complex solutions than the axial rod. First, a 12-inch aluminum beam is cantilevered and subject to a vertical tip load of 1000 lb as shown in Figure 4.7a. The analytical solution for displacement is a third



(a) Tip Load



(b) Distributed Load

Figure 4.7: Geometry and configuration of a Timoshenko beam with transverse loads.

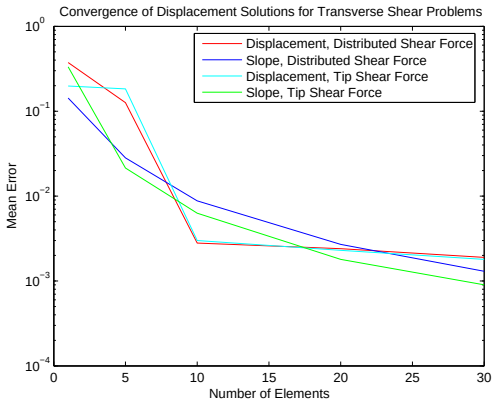
order polynomial. The following equations are the well-known analytical solutions for Euler-Bernoulli beams [25]:

$$\begin{aligned} EI_{yy}u_z''''(x) &= -u_z(x) & u_z'(x) &= \frac{1}{EI_{yy}}\int_0^x M_y(x)dx \\ EI_{yy}u_z'''(x) &= V_z(x) & u_z(x) &= \frac{1}{EI_{yy}}\int_0^x \int_0^x M_y(x)dx\,dx \\ EI_{yy}u_z''(x) &= M_y(x) \end{aligned} \tag{4.5}$$

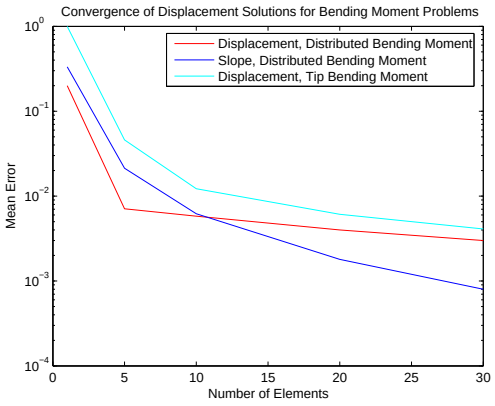
Additional terms are required to include the translational displacement due to shear forces (which is assumed to be zero in Euler-Bernoulli beams), and when combined these yield equations for a Timoshenko beam.

$$\begin{aligned} F_z(x) &= F_z(L) & M_y(x) &= M_y(0) - \int_0^x F_z(x)\,dx_1 \\ \theta_y(x) &= \frac{F_z(L)x}{2EI_{yy}}(2L - x) & u_z(x) &= \frac{F_z(L)x^2}{6EI_{yy}}(3L - x) + \frac{F_z(L)x}{AG} \end{aligned} \tag{4.6}$$

Based on the choice of constitutive equations, the algorithm can model both types of beams. It involves choosing the effective area (K) in the two shear directions (zero for Euler-Bernoulli). Because this is a mixed formulation, the matrices containing the



(a) Transverse Shear



(b) Bending Moment

Figure 4.8: Mean error versus number of elements for Timoshenko beams.

constitutive and inertia properties need not be invertible, and thus a zero shear area may be specified.

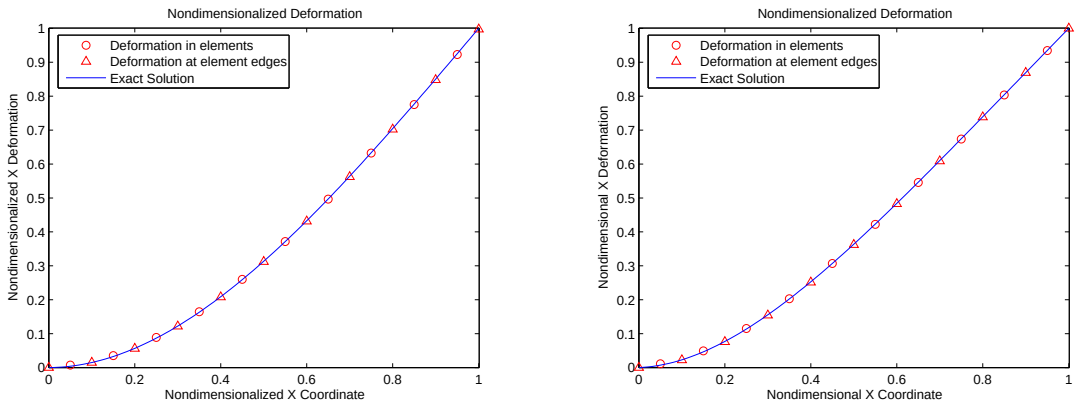
As the number of elements is increased, the result approaches the exact solution. Figure 4.8 presents the convergence graphically. Less than one percent error remains with a mesh of 10 elements, using the displacement at the element centroids as the convergence criteria (Figure 4.9a). Convergence for all four Timoshenko beam configurations is displayed in Table 4.3.

Next, the tip load at the free end is removed and a distributed shear load of 100 lb/in is applied along the perpendicular axis as shown in Figure 4.7b. The analytical solution is given by the following equations:

$$
\begin{aligned}
F_y(x) &= F_y(L) - v_y(x)x \\
M_z(x) &= M_z(0) - \int_0^x F_y(x)\,dx \\
\theta_z(x) &= \frac{v_y x}{6EI_{zz}}\left(x^2 - 3Lx + 3L^2\right) \\
u_y(x) &= \frac{v_y x^2}{24EI_{zz}}\left(x^2 - 4Lx + 6L^2\right) + \int_0^x \frac{F_y(x)}{AG}dx
\end{aligned}
\tag{4.7}
$$

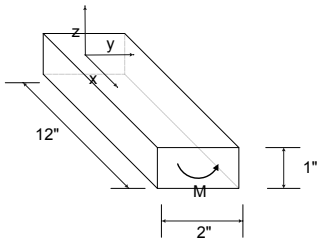
Table 4.3: Convergence of Timoshenko Beam Problems

Loading Condition		Mean Error for Each Mesh				
		1	5	10	20	30
Bending Moment (Distributed)	$\bar{u}$	0.2000	0.0071	0.0058	0.0040	0.0030
	$\hat{u}$	0.2500	0.0180	0.0056	0.0017	0.0008
	$\bar{\rho}$	0.3333	0.0213	0.0062	0.0018	0.0008
	$\hat{\rho}$	0.0000	0.0000	0.0000	0.0000	0.0000
Bending Moment (Tip)	$\bar{u}$	1.0000	0.0460	0.0122	0.0061	0.0041
	$\hat{u}$	0.0000	0.0000	0.0000	0.0000	0.0000
	$\bar{\rho}$	0.0000	0.0000	0.0000	0.0000	0.0000
	$\hat{\rho}$	0.0000	0.0000	0.0000	0.0000	0.0000
Transverse Load (Distributed)	$\bar{u}$	0.3743	0.1260	0.0028	0.0024	0.0019
	$\hat{u}$	0.0000	0.0195	0.0066	0.0020	0.0010
	$\bar{\rho}$	0.1429	0.0282	0.0088	0.0027	0.0013
	$\hat{\rho}$	0.5000	0.0135	0.0026	0.0008	0.0003
Transverse Load (Tip)	$\bar{u}$	0.1979	0.1834	0.0030	0.0023	0.0018
	$\hat{u}$	0.2492	0.0181	0.0057	0.0019	0.0010
	$\bar{\rho}$	0.3336	0.0214	0.0063	0.0018	0.0009
	$\hat{\rho}$	0.0004	0.0000	0.0000	0.0000	0.0000

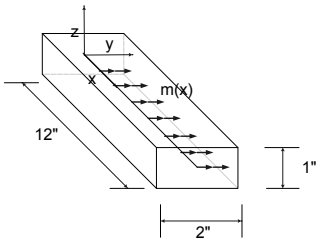


(a) Tip Load Configuration (b) Distributed Load Configuration

Figure 4.9: Nondimensionalized deformation of a Timoshenko beam with transverse loads.



(a) Tip Moment



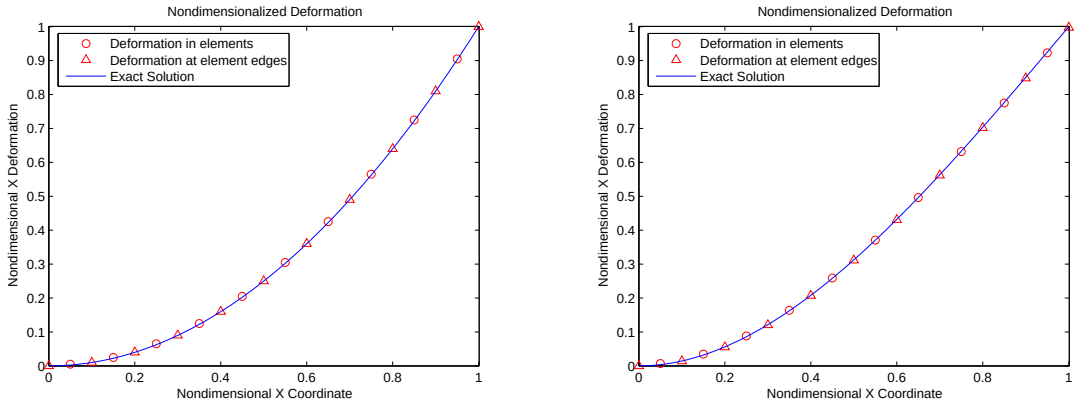
(b) Distributed Moment

Figure 4.10: Geometry and configuration of a Timoshenko beam with bending moments.

For this case the displacement solution is a fourth-order polynomial, which is the highest order for this class of problems. The data is accurate within a percent using a mesh of 10 elements (Figure 4.9b).

The same beam is also loaded with a 1000 in-lb bending moment at the tip (Figure 4.10a) and a distributed moment of 100 in-lb/in (Figure 4.10b). For a tip moment, the exact equations are as follows:

$$\begin{aligned} F_y(x) &= 0 \\ M_z(x) &= M_z(0) \\ \theta_z(x) &= \frac{M_z(0)x}{EI_{zz}} \\ u_y(x) &= \frac{M_z(0)x^2}{2EI_{zz}} \end{aligned} \tag{4.8}$$



(a) Tip Moment Configuration

(b) Distributed Moment Configuration

Figure 4.11: Nondimensionalized deformation of a Timoshenko beam with bending moments.

For a distributed moment, the order of the solution is increased and the following relationships are used:

$$\begin{aligned} F_y(x) &= 0 \\ M_z(x) &= M_z(0) - \int_0^x m_z(x)dx \\ \theta_z(x) &= \frac{m_z x}{2EI_{zz}}(2L - x) \\ u_y(x) &= \frac{m_z x^2}{6EI_{zz}}(3L - x) \end{aligned} \tag{4.9}$$

In each configuration the resultant force and force strains are zero. In both cases accurate results are achieved with 10 elements, as depicted in Figure 4.11.

The tests with Timoshenko beams illustrate the algorithm's capability to handle simple beam configurations and loading. While additional boundary conditions (such as simply supported) are not included here, the analytical solutions are of the same order and should be expected to converge in the same manner.

4.3 Cantilevered Elastica

The cantilevered elastica is used to test the algorithm against a highly nonlinear problem. Hopkins and Ormiston [30] use it to demonstrate that linear elements, when constrained to move with their "parent" elements, can be combined to capture nonlinear deformations in beams. Elastica is the shape of the elastic curve that comes from solving the differential equation for a beam with large deflections [25]:

$$\frac{M}{EI} = \frac{\frac{\partial^2 v}{\partial x^2}}{[1 + \left(\frac{\partial v}{\partial x}\right)^2]^{3/2}} \tag{4.10}$$

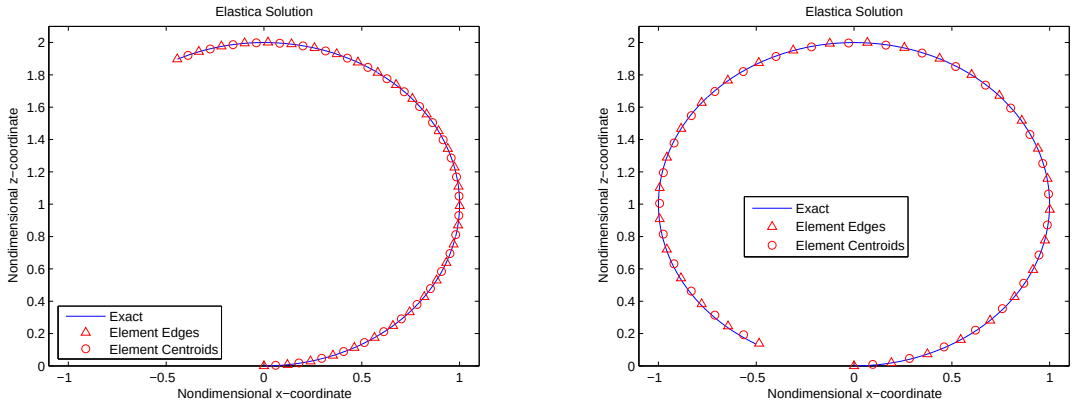
This relationship provides an analytical basis for comparison. For this case the beam and loading configuration from Figure 4.10a are used, and the modulus of elasticity is decreased to 10x10^3 psi. This makes the beam very elastic and it deforms easily under small loads. First, a 500 in-lb tip moment is applied, deflecting the beam back past its root. Next, an 800 in-lb moment is applied to curl the beam into a nearly complete circle. Results for 30 elements are displayed in Figure 4.12, with the deformed coordinates nondimensionalized by the radius of curvature.

In this problem, an initial guess of zero for the solution is not satisfactory, and does not always reach a solution. This is a unique problem with nonlinear equation solvers which must be addressed. An incremental technique is used (discussed in chapter 3), dividing the load into ten equal steps and updating the initial guess after each increment is solved. In this manner the initial guess is never very far from the actual solution.

Because the applied load is a tip moment, the curvature (ρ) is a constant value, forming a circle that satisfies the following equation:

$$\frac{M}{EI} = \frac{1}{\rho} \tag{4.11}$$

This gives an exact solution for comparison. For other loading conditions (such as a tip load), the equations are transcendental and must be solved numerically. To



(a) 500 in-lb tip moment

(b) 800 in-lb tip moment

Figure 4.12: Cantilevered elastica results for a tip moment.

Table 4.4: Summary of Convergence for the Cantilevered Elastica Problem

Loading Condition		Mean Error For Each Mesh				
		5	10	20	30	40
Tip Moment 500in-lb	$\bar{u}$	0.4586	0.0388	0.0214	0.0093	0.0067
	$\hat{u}$	0.0721	0.0177	0.0038	0.0024	0.0017
	$\bar{\theta}$	0.0122	0.0023	0.0005	0.0002	0.0001
	$\hat{\theta}$	0.0027	0.0005	0.0001	0.0000	0.0000
Tip Moment 800in-lb	$\bar{u}$	1.7719	0.4195	0.0260	0.0152	0.0095
	$\hat{u}$	3.1515	0.2880	0.0222	0.0066	0.0040
	$\bar{\theta}$	0.2957	0.0270	0.0027	0.0011	0.0006
	$\hat{\theta}$	0.3702	0.0303	0.0019	0.0004	0.0002

evaluate convergence, the x-z coordinates of the solution are compared with those of the exact solution (4.11), and the mean error is computed using equation (4.2). Figure 4.13 presents the graphical convergence of the solutions.

In the most extreme case, the mean error in $\bar{u}$ at the centroids is less than a percent with 40 elements. Table 4.4 depicts the convergence for both loading conditions. While the scenario of curling a beam into a circle does not fit the application of a wing or rotor blade, it demonstrates effectiveness for an exaggerated case and places confidence in the algorithm's ability to handle highly nonlinear problems.

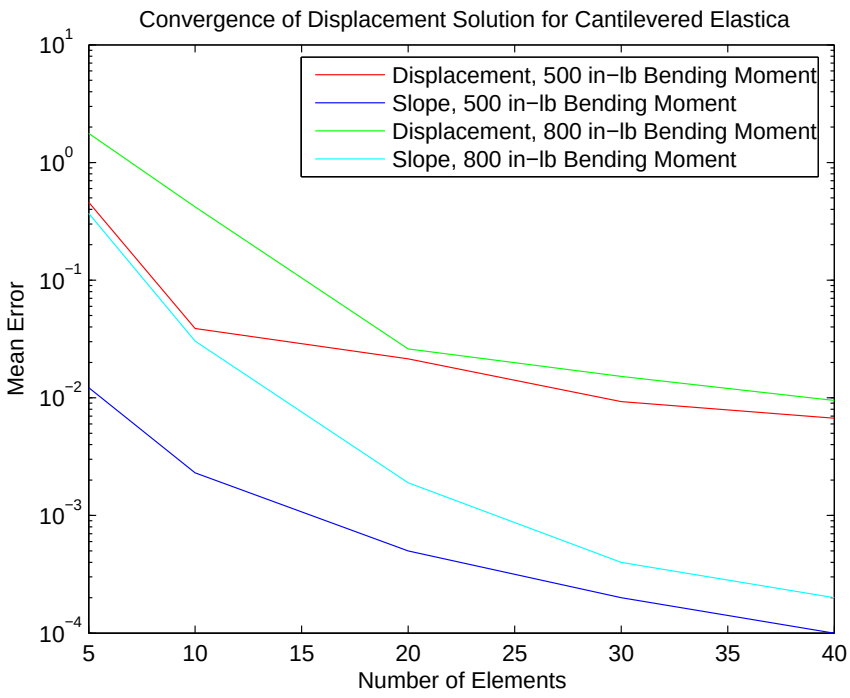
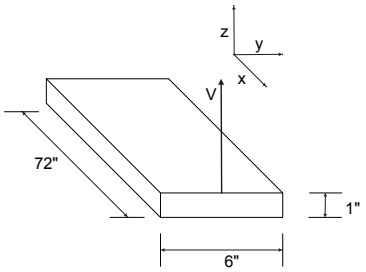


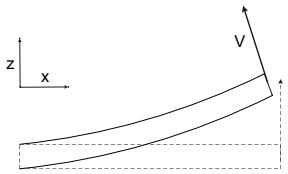
Figure 4.13: Mean error versus number of elements for the Cantilevered Elastica.

4.4 Beam with a Follower Force

The final static problem involves a cantilevered beam with a follower force at the tip as depicted in Figure 4.14. Follower forces are typical of aerodynamic loads, which are functions of the body's orientation. As the body deforms, the loads move with it. In this algorithm the applied loads are specified in the B frame (the deformed frame), and they move with the beam as it deforms. This is not obvious with small displacements, but becomes evident as the deformation increases. In this problem the nonlinear effects of a follower force are captured. Here, the aluminum beam is 72 inches long with a 6x1 inch cross section, and the applied load is 1 kip. By using kips as the basic unit of force, the state vector is scaled well and the algorithm runs efficiently and quickly. The tip load changes its orientation to stay perpendicular to the beam, making it a follower force. In order to designate a force that is not a follower force, it must be premultiplied by C^{bB} at each time interval. This is possible



(a) Geometry and Loading



(b) Follower Force

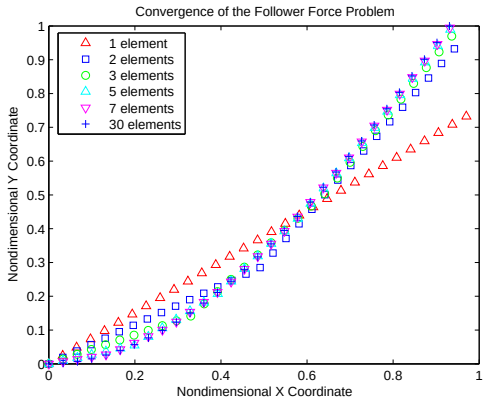
Figure 4.14: The follower force is implemented as a tip load that stays perpendicular to the surface as the beam deforms.

but adds complexity to the system equations. It is required, however, for certain forces such as those associated with gravitational or magnetic fields.

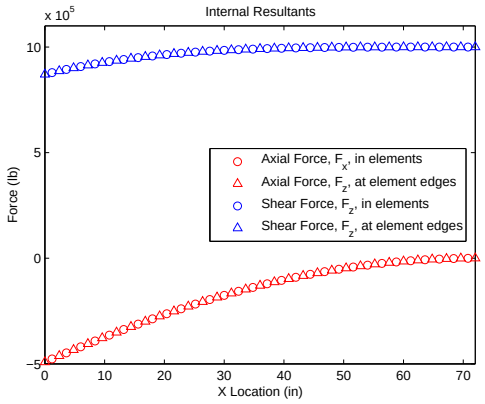
At this point, the algorithm has demonstrated sufficient convergence for the most extreme cases of elastica. Therefore, it is assumed convergence will also occur in this case. The problem is solved several times, each with a finer mesh, and the percent change from one displacement solution to the next is recorded in Table 4.5. At 30 elements, the percent change is less than one. Figure 4.15a illustrates the convergence of the displacement solution for several meshes. Solutions are nondimensionalized by the maximum value of the finest mesh for comparison. Note that in Figure 4.15b the internal shear force is no longer constant throughout the beam. This is due to the change in orientation of the force and the large rotation of the beam. Recall that the values for internal force are given in the deformed frame. With large deformations, the orientation of this frame changes significantly along the length of the beam. At the beam root, the reaction is equal to the z component of the applied load after deformation. As the x-coordinate approaches the end of the beam, this force increases in magnitude, approaching the value of the applied load. The complete solution for all variables is located in Appendix A.

Table 4.5: Summary of Convergence for the Follower Force Problem

Elements	Mean Change in Displacement
2	0.2502
3	0.1287
5	0.1212
7	0.0650
10	0.0526
20	0.0090
30	0.0063



(a) Displacement for Several Meshes



(b) Internal Forces with 30 elements

Figure 4.15: Results for a cantilevered beam with a transverse follower force at the tip.

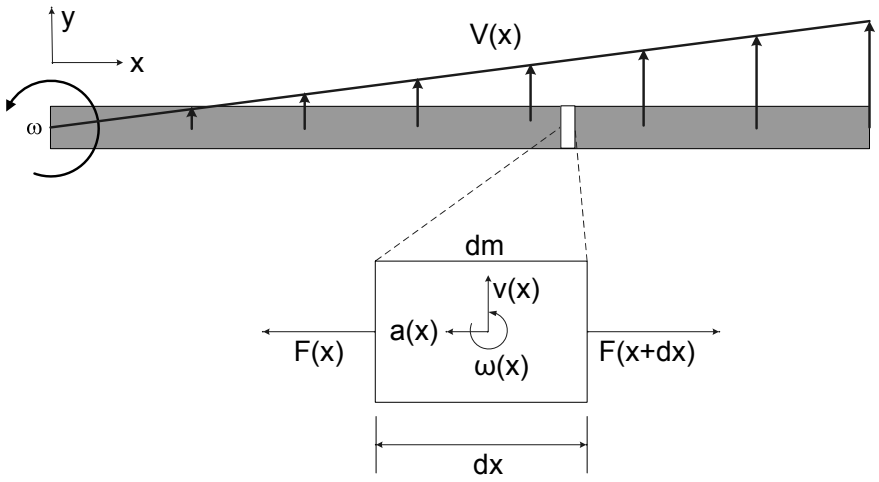


Figure 4.16: A steady rotating beam experiences deformation due to inertial forces.

4.5 Beam Rotating at a Constant Angular Velocity

While all previous problems have been in static equilibrium, this algorithm is also capable of solving steady problems that are in dynamic equilibrium. For problems that do not begin at rest, a complete set of initial conditions is required ($\hat{u}_1$, $\hat{\rho}_1$, $\hat{P}_1$, $\hat{H}_1$). Some of these variables may be unknown, and therefore the steady-state solver must be used first to determine their values. The constraint imposed by equation (3.106) is not that velocities and momenta be zero, but that they are time invariant. In other words, it is a steady-state condition rather than a static condition. To demonstrate this, the algorithm is applied to a beam rotating at a constant angular velocity. The beam is cantilevered to a hub, which rotates at a constant rate. The velocities v_b and ω_b are specified for each element to generate a beam which is rotating at a constant velocity about its root with fixed-free boundary conditions. In this configuration the beam will experience elongation due to the inertial forces which hold it in place, as illustrated in Figure 4.16.

For this problem, the centripetal acceleration of a point in the deformed beam is given by:

$$a(x) = [r(x) + u(x)]\omega^2 \tag{4.12}$$

Integrating the acceleration over the length of the beam yields the following expression:

$$\begin{aligned} F(x) - F(x+dx) &= \int_0^\ell a(x)dm \\ F(x+dx) &= F(x) - \int_0^\ell [r(x) + u(x)]\omega^2 m\, dx \\ F(x) &= F(0) - \int_0^x [r(x) + u(x)]\omega^2 m, dx \end{aligned} \tag{4.13}$$

The internal force at the root, $F(0)$, is determined by solving equation (4.13) for the free tip condition, where $F(\ell) = 0$:

$$F(0) = F(\ell) + \int_0^\ell [r(x) + u(x)]\omega^2 m\, dx \tag{4.14}$$

This relationship is substituted for $F(0)$ to obtain the following:

$$\begin{aligned} F(x) &= \int_0^\ell [r(x) + u(x)]\omega^2 m\, dx - \int_0^x [r(x) + u(x)]\omega^2 m\, dx \\ F(x) &= \int_x^\ell [r(x) + u(x)]\omega^2 m\, dx \end{aligned} \tag{4.15}$$

For axial deformation due to the internal forces, the following equation is applied:

$$\begin{aligned} u(x) &= \int_0^x \gamma(x)dx \\ u(x) &= \frac{1}{AE}\int_0^x F(x)\, dx \end{aligned} \tag{4.16}$$

Clearly, the relationship formed between (4.16) and (4.15) is transcendental. As deformation increases, the internal force increases which makes the deformation larger. For stiff solids, where the deformation is small, this increase in internal force is neg-

ligible. The internal force can therefore be approximated by treating the beam as a rigid body:

$$\begin{aligned} a(x) &\approx r(x)\omega^2 \\ F(0) &\approx \frac{1}{2}m\ell^2\omega^2 \\ F(x) &\approx \frac{1}{2}m\omega^2(\ell^2 - x^2) \end{aligned} \tag{4.17}$$

After approximating the internal force, equation (4.16) is used to determine the deformation.

$$\begin{aligned} u(x) &= \frac{1}{AE}\int_0^x F(x)\,dx \\ u(x) &= \frac{m\omega^2}{2AE}\left(\ell^2 x - \frac{x^3}{3}\right) \end{aligned} \tag{4.18}$$

This provides a fast method for checking data to ensure it is converging to the correct solution when the material is known to be stiff and experience only small deformations. The alternative is to numerically solve equations (4.15) and (4.16) simultaneously. This was not done, however, and the convergence of the solution was based on the percent change in deformation between finer meshes (as in the previous section). When this percent change is sufficiently small (in this case less than a tenth of a percent), the solution is converged. The results are also compared to the approximation from equations (4.17) and (4.18), which proves to work well for aluminum beams but is not sufficient for very elastic solids.

Six cases are investigated for this configuration, including three different geometries rotating at both 3Hz and 20Hz. The first geometry is the 12-inch aluminum beam from Figure 4.7. The second geometry is the 72-inch beam from Figure 4.14. For the final geometry, the modulus of elasticity from the 12-inch beam is decreased to $10\text{x}10^2$ psi, one order of magnitude smaller than the elastica problem. In this case the deformation and internal forces are greater than the rigid approximation, particularly at high speeds. This was expected, because the material is not stiff. Table 4.6

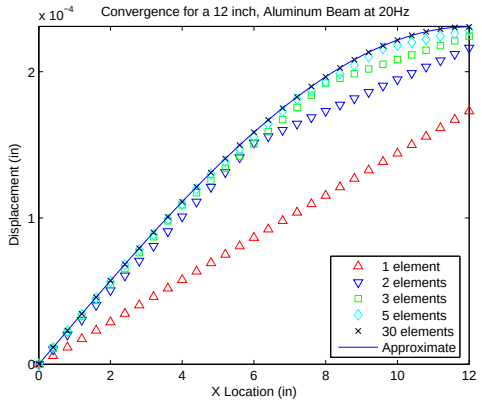
Table 4.6: Summary of Convergence for Steady Rotation Problems

Configuration		Mean Difference From Previous Mesh						
Geometry	Speed	2	3	5	7	10	20	30
12 inch	3 Hz	0.5000	0.0758	0.0355	0.0113	0.0058	0.0029	0.0007
	20 Hz	0.5101	0.0773	0.0363	0.0115	0.0060	0.0030	0.0007
12 inch elastic	3 Hz	0.5002	0.0762	0.0357	0.0113	0.0059	0.0029	0.0007
	20 Hz	0.6254	0.0946	0.0451	0.0139	0.0072	0.0039	0.0009
72 inch	3 Hz	0.5000	0.0758	0.0355	0.0113	0.0058	0.0029	0.0007
	20 Hz	0.5004	0.0759	0.0355	0.0113	0.0058	0.0029	0.0007

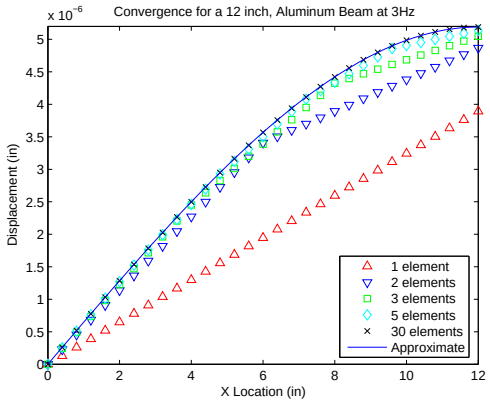
shows the percent change for each of seven successive meshes in each scenario. At 30 elements, the solution is fully converged for all cases. It is interesting to note that all six cases have approximately the same percent change among the different meshes, since they are all modeling a second-order function.

This was the first problem where scaling difficulties were encountered. Ultimately, solutions for long beams, large meshes, and high velocities were unattainable with standard units of pounds, inches, and seconds. After investigating several approaches to rescaling the state vector, successful convergence was achieved when the basic unit of time was changed to milliseconds. This also increased the basic unit of force to 10^6 pounds, or megapounds. Fortunately, this caused all field variables to appear on the order of 10^{-3}, and the solution converged quickly. The exception to this occurred for the elastic beam at 3 Hz with 10 or more elements. Here it was necessary to switch back to standard units to obtain the correct solution. Since the speed and modulus are both very small in this case, using large units for force is not appropriate. It demonstrated that there is not one perfectly robust set of units for all problems, but that the user must know in advance the correct scaling to use.

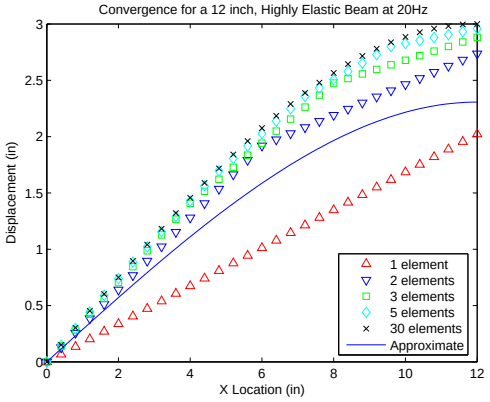
Figure 4.17 shows the progression of the solution as the number of elements is increased. The approximate solution from equations (4.17) and (4.18) is also displayed. The small deformations show that the approximation is good except for the elastic case at high speed. In this case the model converges to a numerical solution that takes into account the transcendental relationship between equations (4.15) and



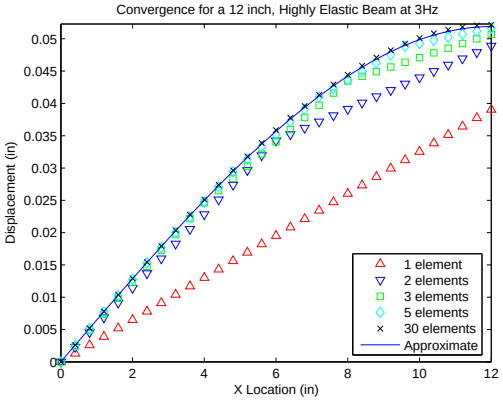
(a) 12-in Aluminum, 20Hz



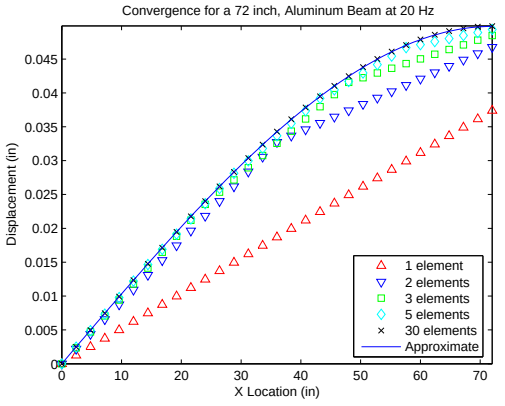
(b) 12-in Aluminum, 3Hz



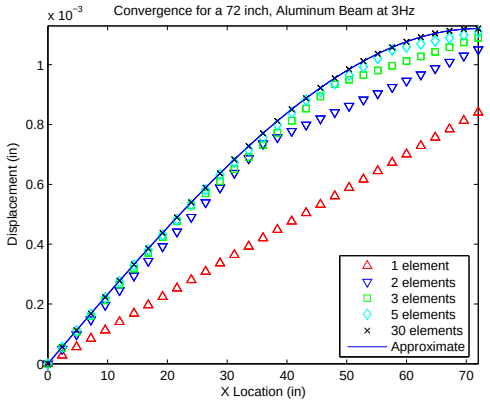
(c) 12-in Elastic, 20Hz



(d) 12-in Elastic, 3Hz



(e) 72-in Aluminum, 20Hz



(f) 72-in Aluminum, 3Hz

Figure 4.17: Convergence for a cantilevered, rotating beam.

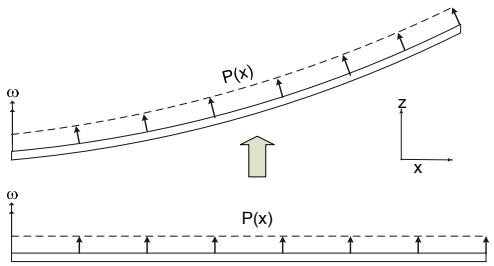


Figure 4.18: Rotating beam with a constant, distributed load.

(4.16). The displacements and forces are higher than would be estimated by treating the beam as a rigid body. This is an excellent example of an application for this type of solver, when modeling a system with rigid bodies is no longer a valid assumption. Complete solutions are found in Appendix A.

4.6 Rotating Beam with Applied Loads

The final steady-state problem combines loading and prescribed motion. The geometry used is the same as in the previous section. For this scenario, a distributed transverse load of 50 lb/in is applied over the length of the beam, as shown in Figure 4.18. The load remains in the deformed coordinate system (a follower force) as the beam deforms. First, the solution is obtained for a stationary beam. Next, a constant angular velocity of 3Hz is applied about the z-axis, and the beam is loaded again while rotating. Both cases are evaluated with the steady-state solver, and the converged results are compared.

This problem is similar to a rotor blade in hover, with the exception that the applied load is constant rather than elliptical. This is done for simplicity, though with enough elements, an elliptical distribution can be accurately represented. Convergence is based on the percent change in the displacement solutions, as it was in the steady rotation problem. It is assumed that the algorithm will converge to the exact solution, which is unknown. This assumption is based on performance with previous problems

Table 4.7: Mean Change in the Displacement Solution for a Rotating Beam with a Distributed Load

Elements	Stationary	Rotating
10	0.0952	0.0991
20	0.0083	0.0143
30	0.0056	0.0083

(*i.e.* the elastica problem), where the solution was within a percent of the exact and the displacement error continued to decrease as more elements were used. Table 4.7 presents the percent change in each solution for successive meshes of 5, 10, 20, and 30 elements. The beam experiences greater deformation while stationary than when rotating. The difference between the two solutions is due to the the stiffening effect of the rotation (Figure 4.19). The beam's inertia generates an axial force, and this tension reduces the amount of vertical displacement.

Units of megapounds, inches, and milliseconds were used for this test case, since this was required for the problem of steady rotation. However, convergence of the optimization algorithm was significantly slower with the applied load. While using kips, inches, and seconds was more efficient for the stationary problem, it was not as effective with the added rotation, particularly for large meshes. This illustrates once again the scaling problem and the necessity of choosing the correct units in order for the algorithm to converge. The complete solutions for both configurations are located in Appendix A.

4.7 Axial Vibration of Rods

Having tested the algorithm against several static and steady motion problems, the remaining cases introduce time accurate scenarios by removing the constraint imposed by equation (3.106). The steady-state algorithm is applied first to obtain the full set of initial conditions, which are introduced back into the equations to obtain the solution at the first time step. The results from each time step become initial conditions for the subsequent time steps.

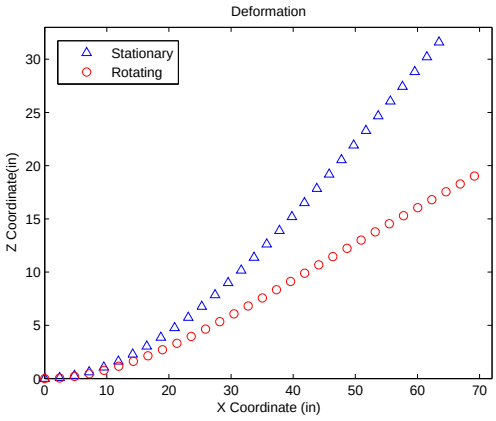


Figure 4.19: Displacement of a rotating beam with a distributed load

First, the algorithm is used to simulate the axial vibration of a fixed-free rod. The fundamental frequency for a cantilevered rod is determined from the following equation [17]:

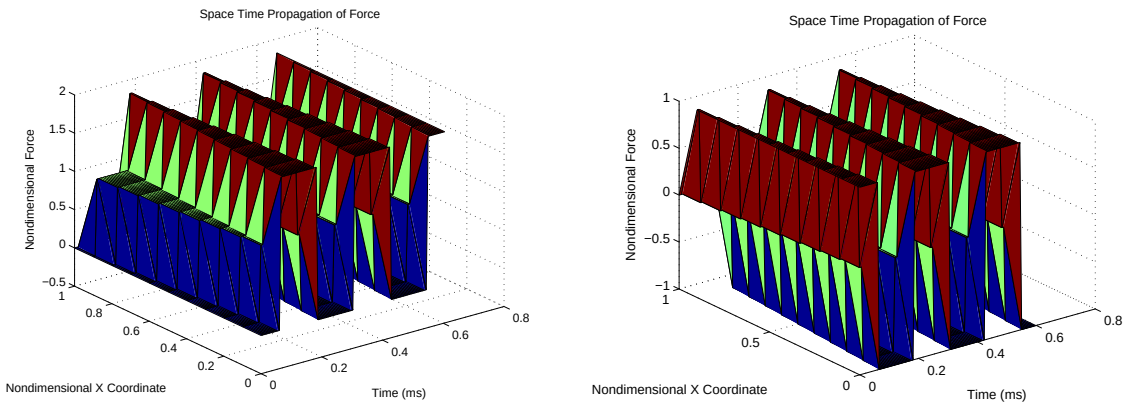
$$\omega_n = \frac{\pi}{2L}\sqrt{\frac{E}{\rho}} \tag{4.19}$$

Equations (3.92 - 3.105) reduce to a linear system for axial problems. Atilgan and Hodges solved this system of equations for problems of wave propagation in rods [3]. When the problem is linear, solutions are obtained by assembling and solving a simple linear algebra problem ($Ax = b$) at each time step. Initially, the problem was solved using this method to reproduce results from the literature. Afterwards, the nonlinear algorithm was applied and found to yield the same solution in every case.

Standard finite element models perform modal analysis on structures by solving the following eigenvalue problem [17]:

$$\left[K - \omega_n^2 M\right] D = 0 \tag{4.20}$$

This is accomplished using an element mass matrix (K) and stiffness matrix (M), with the degrees of freedom in D constrained to enforce the boundary conditions. The formulation of the mass matrix determines whether the estimate for the natural



(a) Step Input applied at t=0

(b) Load Released from Static Equilibrium

Figure 4.20: Propagation of force through a fixed-free rod with $\Delta x/\Delta t = \sqrt{E/\rho}$.

frequency will be high or low. Because the algorithm used in this research does not work with differential equations, it is not suited to solve the eigenvalue problem for modal analysis. The frequency response must be obtained from a Fast Fourier Transform (FFT) of the time history.

Consider the fixed-free rod in Figure 4.1 with a 100 lb axial load applied at the tip. The system is initially in static equilibrium. At t=0, the load is released, and the beam's deformation is measured in the axial direction over time. A similar response is obtained when the beam begins at rest and a "step" input is applied (the 100 lb load is instantaneously applied at t=0). The difference is in the equilibrium position about which it oscillates. The step input oscillates about the static equilibrium point, while releasing the load causes it to oscillate about zero. For all of the remaining cases, the load will be applied initially and then removed. Figure 4.20 shows the propagation of the wave in space and time for each case. The forces and x coordinates are nondimensionalized for comparison with the literature [3].

Testing the algorithm on this problem reveals important information about element scaling for dynamic problems. When the element aspect ratio, $\Delta x/\Delta t$, is equivalent to the wave speed, $\sqrt{E/\rho}$, the beam vibrates at the exact fundamental frequency. This holds true regardless of the number of spatial or temporal elements,

and is accomplished because the wave is able to propagate through the medium at the speed of sound. This is depicted in Figure 4.20 as the wave travels through a single element per time step, moving at the exact speed of sound, which results in an oscillation at the natural frequency. However, if the ratio $\Delta x/\Delta t$ is not exactly the same as the wave speed, noise appears in the solution which can cause the fundamental frequency to vary. Figure 4.21 demonstrates this effect using several different time steps for a single element. The time step is adjusted such that the element aspect ratio is tested both above and below the wave speed.

For the fixed-free rod in Figure 4.1, equation (4.19) yields a fundamental frequency of 4137 Hz. With 10,000 iterations of time data, the maximum number of FFTs allowed is 8192 (it must be a power of 2). Applying an FFT to the time response, the natural frequency of the rod is determined numerically (see Appendix D for details). When the aspect ratio is equivalent to the wave speed, the peak frequency occurs at the expected natural frequency for the rod. However, this frequency appears to increase as the time step becomes smaller. This is alarming because in general one expects accuracy to improve as the step size is decreased for initial value problems.

Bauchau [11] discusses problems encountered with high frequency noise creeping into solutions for multibody systems, and implements an energy decaying algorithm which dissipates this noise to obtain correct solutions. Whether this phenomenon is the same is unknown. When the solution is obtained using larger meshes of 10 or 20 elements, a shift in the peak frequency does not occur when the time step is varied. This is illustrated in Figure 4.22. The noise is clear in the time response, but frequency analysis reveals that for multiple elements the fundamental frequency is relatively unchanged. They all have a peak at the expected natural frequency for the rod. The noise observed in the time history for each solution does not shift this peak significantly, only make it broader or narrower. The exact cause of the frequency shift for a single element is unknown. Application of an energy decaying algorithm

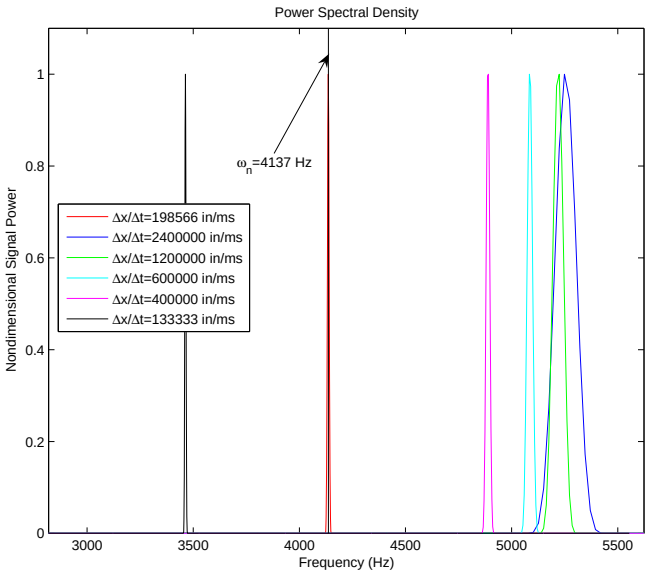


Figure 4.21: Effect of $\Delta x/\Delta t$ on the axial vibration with one element.

may correct the problem, but at this time the algorithm has not yet been developed for mixed formulations, and is beyond the scope of the current research.

Analysis of the time response for various element sizes reveals further effects of the noise. When $\Delta x/\Delta t$ is the same as the wave speed, the forces, momenta, and velocities appear as square waves. The displacement field follows a triangular wave pattern (see Figure 4.23). This indicates abrupt changes in velocity (corresponding to infinite accelerations) which would not occur. This is a result of using constant shape functions for the field quantities. The wave discretely steps through the rod rather than moving fluidly as through a continuum. The actual response should be a sine wave. Also, the sampling rate is equal to twice the frequency of vibration, which in general would result in poor resolution. Oddly, it gives the exact solution for one element, whereas solutions with smaller time steps are incorrect. For example, if more sampling points were added to Figure 4.23, one might expect it to appear more like a sine wave and be closer to the correct frequency. Instead, the frequency is driven higher, which is the incorrect solution.

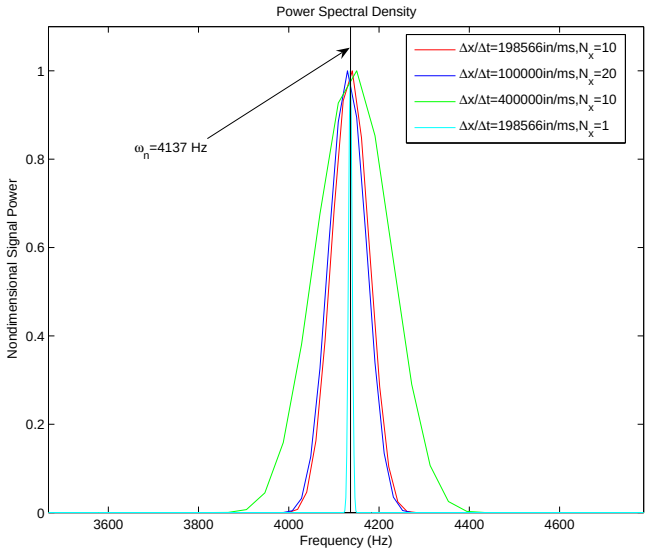


Figure 4.22: Effect of varying $\Delta x/\Delta t$ on the axial vibration of a rod with multiple elements.

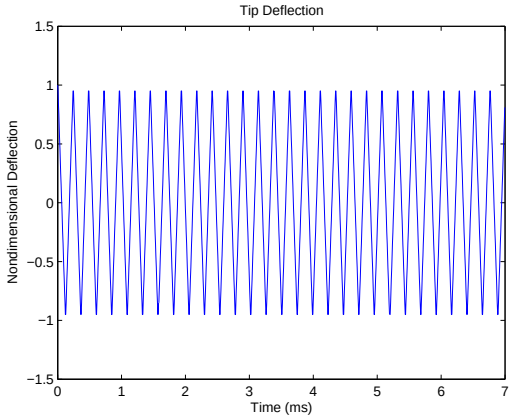
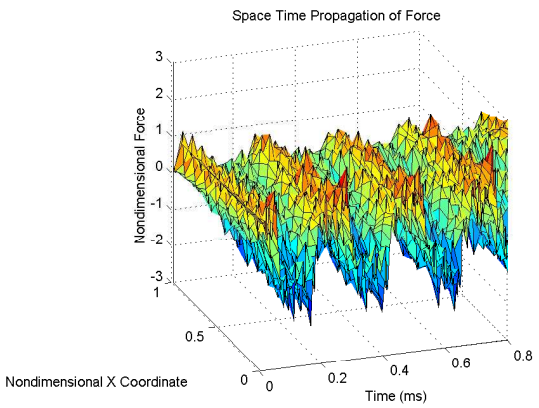
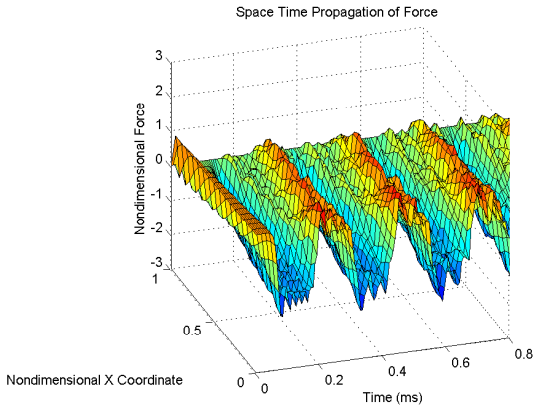


Figure 4.23: Tip displacement for a rod vibrating at its natural frequency with $\Delta x/\Delta t = \sqrt{E/\rho}$, $N_x = 1$.



(a) 10 Elements, $\Delta t = 0.003ms$



(b) 20 Elements, $\Delta t = 0.006ms$

Figure 4.24: Space time propagation of force through a fixed-free rod when $\Delta x/\Delta t \neq \sqrt{E/\rho}$.

With multiple elements, the shape of the response becomes noisy and distorted when the time step is not perfectly adjusted to match the wave speed (see Figure 4.24), even when the time step is made sufficiently small to obtain resolution at the expected frequency. This is opposite what one would expect from using a smaller time step, which generally improves resolution of the response and results in a smooth curve. Nevertheless, the frequency response from Figure 4.22 indicates that these rods are still oscillating at their natural frequencies. The conservation of energy is also demonstrated numerically for each case by applying equations (3.58) and (3.59) (summing up the total kinetic and potential energy for each individual time step), demonstrating that total energy remains constant for the full 10,000 iterations (see Figure 4.25). Complete solutions are shown in Appendix B, and are nondimensionalized by their maximum values for simplicity.

4.8 Torsional Vibration of Shafts

Next, the algorithm is used to determine the fundamental frequency of vibration for the same rod in torsion. To perform this, a 100 in-lb torque is initially applied to the rod in static equilibrium. This load is released at t=0, and the time history of

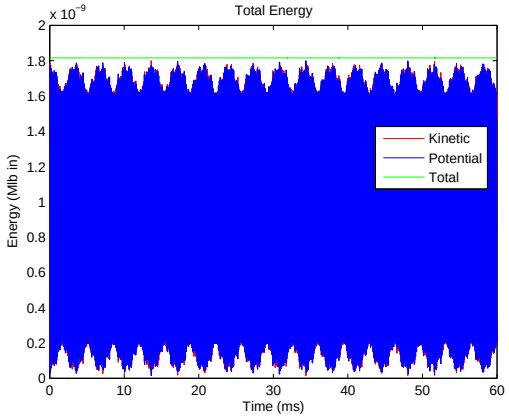


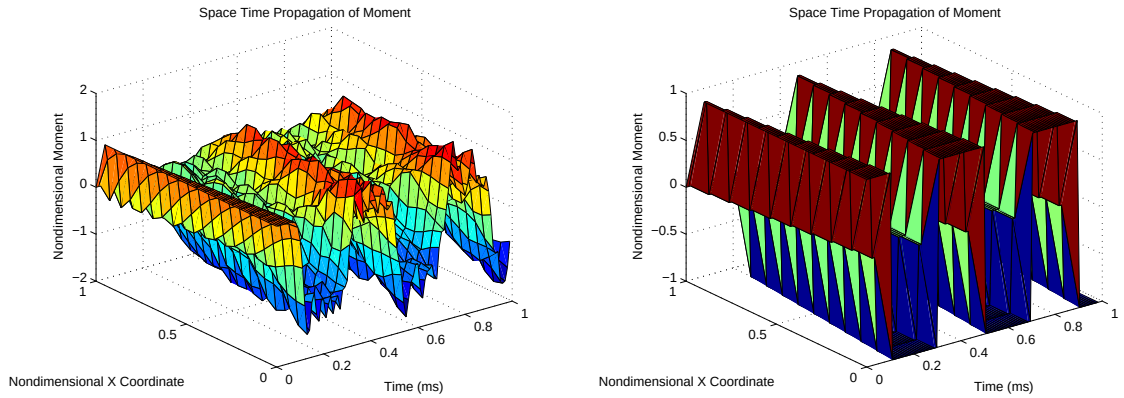
Figure 4.25: Numerical conservation of energy for the vibration of an axial rod, 20 elements, $\Delta t = 0.006ms$

twist and internal torque is recorded. The natural frequency is given by [17]:

$$\omega_n = \frac{\pi}{2L}\sqrt{\frac{G}{\rho}} \tag{4.21}$$

Notice that the shear modulus (G) takes the place of Young's modulus (E) from the axial vibration problem. Otherwise there is no difference. For the 12-in aluminum rod, this natural frequency is 2518 Hz.

First, the ratio $\Delta x/\Delta t$ is set equal to the wave speed, $\sqrt{G/\rho}$, in order to obtain results similar to those in Figure 4.20. Since the solution for both the torsional vibration of shafts and axial vibration of rods is a second-order partial differential equation (PDE), the results are similar. With a single element, the peak frequency is sensitive to the time step used, but when the aspect ratio is equivalent to the wave speed it is correct. With multiple elements, the peak frequency is unchanged. These results are similar in form to the axial vibration problem. Figure 4.26 shows the propagation of the torque through the rod for both the case where the aspect ratio is exactly equal to the wave speed and one case where it is not. For these other cases additional noise appears, but the fundamental frequency is unchanged if multiple elements are used. The frequency response for each simulation is captured in Figure 4.27. The only simulation that did not produce the correct frequency used



(a) 20 Elements, $\Delta t = 0.01ms$ (b) 10 Elements, $\Delta t = 0.0099ms$

Figure 4.26: Space time propagation of torque through a fixed-free rod.

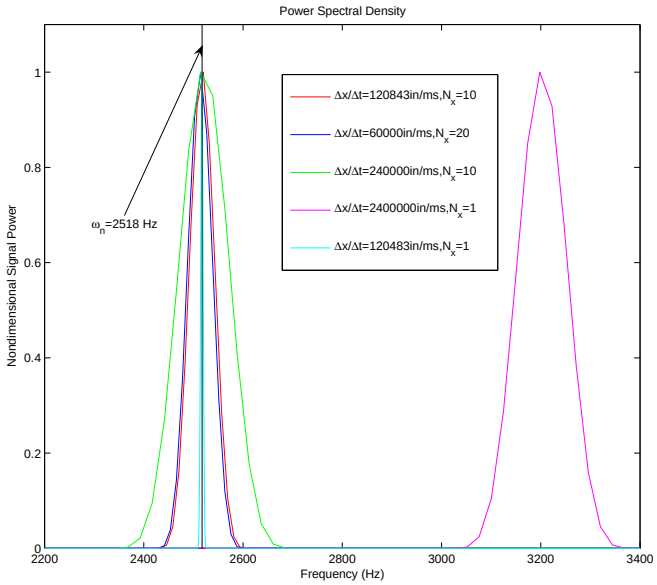


Figure 4.27: Frequency Response for Torsional Vibration with Various Time Steps

a single element and a step size of 0.1 ms. It is interesting to note that by changing the step size to 0.099 ms, the frequency is corrected. Even more interesting is that for a frequency of 2500 Hz one should expect to use a sampling size of no more than 0.04 ms, to provide 10 points per period of oscillation. In this case, the standard rule of thumb does not seem to apply. The necessary sampling rate seems to be dependent on the element size more than the frequency of oscillation. With multiple elements, it converges regardless of the time steps that were used. Complete information from the simulations is presented in Appendix B, and is nondimensionalized by the maximum values.

4.9 Transverse Vibration of Beams

Problems of transverse bending in beams are more complex than axial vibration in rods, involving coupling between the displacements and rotations. The analytical solution is based on a fourth-order PDE. Unlike axial and torsional vibration, there is not a unique wave speed associated with the propagation of forces and moments. The beam will deform axially and transversely at different frequencies. Choices for time steps were made to obtain sufficient resolution to capture the expected natural frequencies. The first bending frequency for a cantilevered beam is calculated with the following equation [17]:

$$\omega_n = 3.516\sqrt{\frac{EI}{mL^3}} \tag{4.22}$$

To excite the bending modes of a beam, it is loaded transversely with a 100 lb tip load in static equilibrium. At t=0, the load is removed and the flapping motion of the beam is simulated. Using the 72-inch aluminum beam from previous problems, the natural frequency for bending about the y axis is 6.2 Hz. Time steps of 1 ms, 16 ms, and 0.145 ms were used. This generates approximately 10 and 160 points per expected period of flapping vibration (for 16 ms and 1 ms), while the third choice gives more resolution in order to capture higher frequency oscillations in the axial direction.

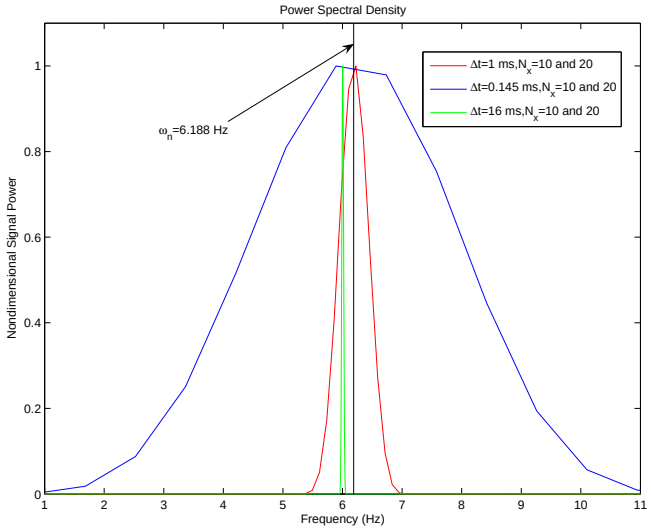
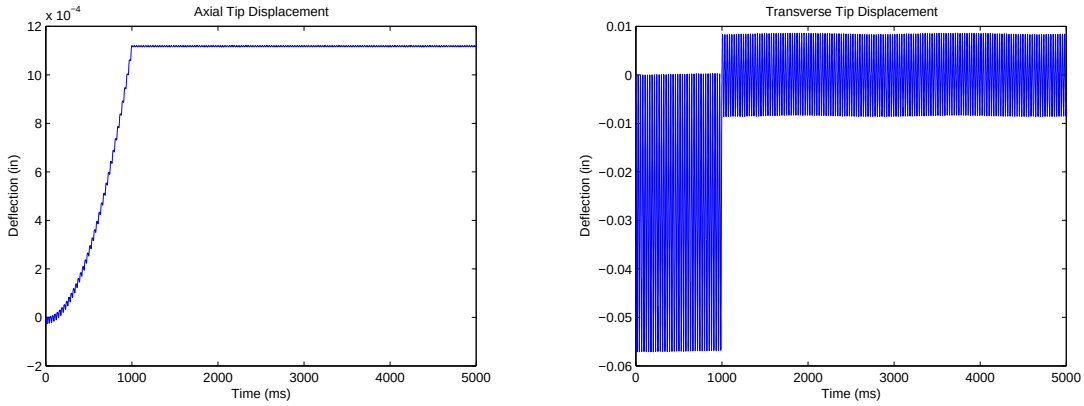


Figure 4.28: Frequency Response for Bending Vibration with Various Time Steps

The frequency analysis of transverse displacement yields the correct natural frequency for bending in that direction. All simulations were performed using either 10 or 20 elements, since this was the minimum for static solutions to converge. The results are depicted in Figure 4.28. All show a peak frequency at the natural bending frequency for the beam. The complete simulation data is contained in Appendix B, and is left in the original units to distinguish the relative magnitudes of axial and transverse forces and displacements.

4.10 Spin Up Maneuver

The spin up is a simple maneuver that is commonly simulated in the literature for multibody dynamics programs [20, 36]. For this maneuver, the 72-inch aluminum beam is attached to a hub (cantilevered) and accelerated from rest to 180 rpm (3Hz) about the z axis. An alternate approach would be to pin the beam and apply a moment about the root, observing the angular displacement. The difference is that the bending vibrations are transmitted to the root as internal forces and moments when the beam is cantilevered. In the response, the beam will lag during the acceleration



(a) Axial Displacement

(b) Transverse Displacement

Figure 4.29: Time history for the spin up of a 72-in aluminum beam, $\Delta t = 0.5ms$, $N_x = 10$

until the beam reaches a steady rotation speed, then vibrate about an equilibrium point at its natural frequency. Axial displacement will also be observed due to the inertial forces. For the simulation, the following angular velocity is prescribed:

$$\omega_b = \begin{cases} 6\pi \left(\frac{5t}{t_f}\right) \ rad/sec, \ t/t_f \leq 0.2 \\ 6\pi \ rad/sec, \ t/t_f > 0.2 \end{cases} \tag{4.23}$$

With $\Delta t = 0.5ms$, $N_x = 10$, and 10000 iterations, this results in a 5 second simulation where the beam is accelerating from 0 to 3 Hz during the first second and then holds a constant angular velocity thereafter. The time history of displacement is depicted in Figure 4.29. Complete results are contained in Appendix C.

The natural lag frequency for the 72-inch aluminum beam as determined by equation (4.22) is 37.12 Hz. Figure 4.30 displays the frequency content of the lateral displacement, and reveals that the fundamental lag frequency of the beam during this maneuver is accurate. The time history also reveals that steady-state values for displacement match those found from solving the steady-state solution of a rotating 72-inch beam at 3 Hz.

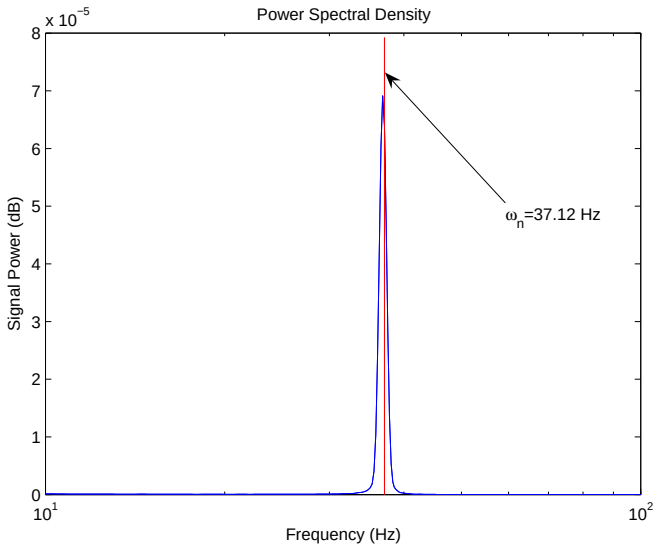
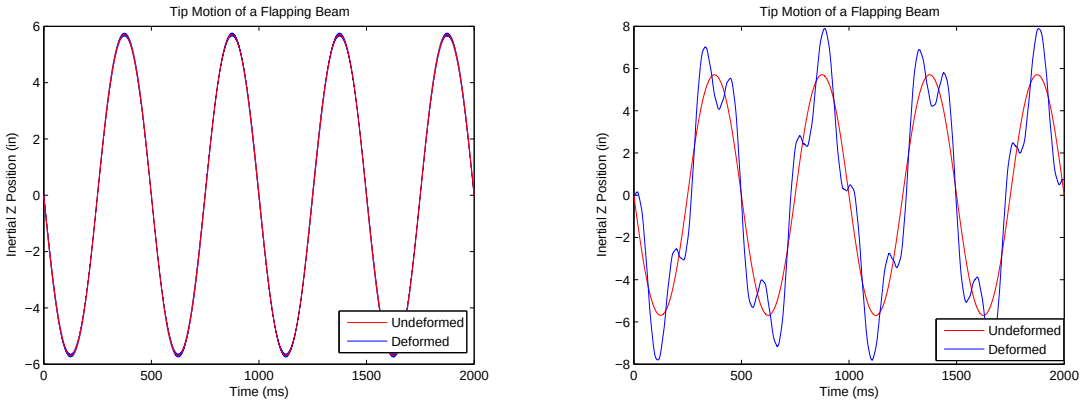


Figure 4.30: Frequency Response of the Spin Up Maneuver for a 72 in aluminum beam, $\Delta t = 0.5ms$, $N_x = 10$

4.11 Flapping Maneuver

To simulate a flapping wing, the 12-inch aluminum beam from Figure 4.7 is used. This beam is cantilevered and given a periodic angular velocity about the y axis. Because it is cantilevered, the mount that it is attached to moves with it. An alternative is to pin the beam and apply a sinusoidal moment. In this case, the beam oscillates at 2 Hz, reaching a maximum of 30 degrees above and below the horizon. This oscillation is forced by prescribing the values for v_b and ω_b, which make up the "rigid" motion of the beam. The same simulation is also performed with the beam from the cantilevered elastica problem. This beam has the same geometry as the 12-inch aluminum beam but the modulus of elasticity is 10x10^3 psi, three orders of magnitude smaller than aluminum. This beam will experience much larger deflections in the flapping motion, and will have a significantly smaller fundamental frequency in bending.

The z-coordinate of the tip is recorded in the inertial frame to demonstrate this motion (see Figure 4.31). This is accomplished by integrating the prescribed velocity (v_b) in the inertial frame to obtain the prescribed position of the tip, then adding



(a) Aluminum Beam, $E = 10e6\,psi$ (b) Highly Elastic Beam, $E = 10e3\,psi$

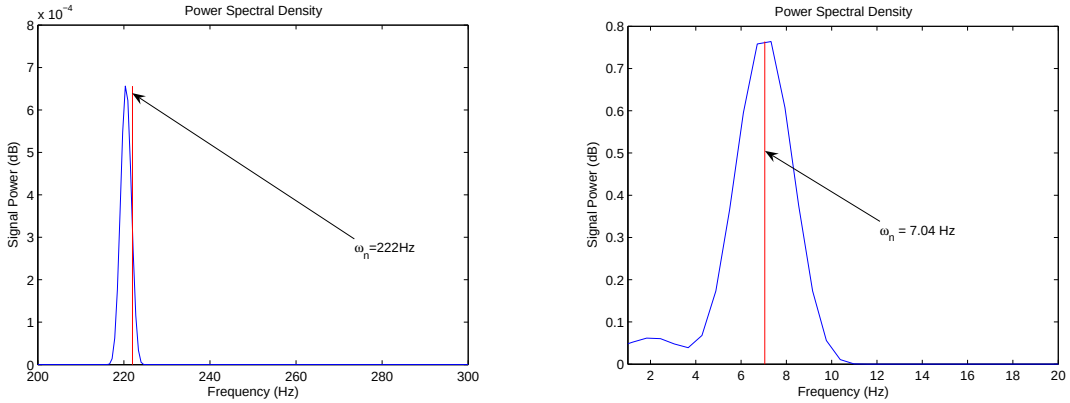
Figure 4.31: Inertial tip response in the flapping maneuver for a 12-in beam,$\Delta t = 0.2ms$, $N_x = 10$

the deformation at each time step. The noise observed is due to the excitation of the flapping frequency of the beam, which is evaluated by equation (4.22) as 222 Hz for aluminum and 7 Hz for the highly elastic beam. A time step of 0.2 ms and 10,000 iterations provides 2 seconds of data, giving four cycles of the flapping motion and providing 22 points of data for each period expected within the aluminum beam's natural flapping frequency. The frequency response in Figure 4.32 indicates the excitation of the flapping frequency in each beam during the maneuver.

In order to set up this maneuver, a sinusoidal angular velocity is prescribed about the y axis. It is the time derivative of the desired flapping motion. To obtain a flapping motion that oscillates at 2 Hz and has a maximum amplitude of 30 degrees, the following relationship is used:

$$\theta_y(t) = \frac{\pi}{6}\sin\left(\frac{2\pi}{1000}t\right) \tag{4.24}$$

In this case t is in milliseconds. The time derivative gives the function for the prescribed velocity, which is calculated to give a discrete value for ω_b at each time interval:



(a) Aluminum Beam, $E = 10e6\,psi$ (b) Highly Elastic Beam, $E = 10e3\,psi$

Figure 4.32: Frequency Response of the Flapping Maneuver for a 12-in beam, $\Delta t = 0.2ms$, $N_x = 10$

$$\omega_b(t) = \frac{\pi^2}{3000}\cos\left(\frac{2\pi}{1000}t\right) \tag{4.25}$$

The beam is discretized by 10 elements in space, which was the minimum to obtain accurate solutions for the Timoshenko beam problems. The frequency response of both beams indicate oscillation at their respective natural frequencies, with the elastic beam experiencing much larger deformations. Figure 4.31 best illustrates the response of the tip, showing both the forced oscillation at 2 Hz and the actual position of the tip which includes the deformation. The aluminum beam experiences very small deformations, while the elastic beam has significant deformation in addition to the prescribed motion. Complete results are contained in Appendix C.

V. Conclusions

5.1 Summary of Results

This research successfully applied HWP to the development of multibody dynamics software for flexible bodies. After deriving Hamilton's Law for beams with closed cross sections (a technique which has been established in the literature [26]), a space-time finite element discretization was performed to generate a system of nonlinear algebraic equations. These equations were solved numerically for common problems in engineering mechanics and structural dynamics. This discretization scheme and solution algorithm provide an alternative to existing multibody dynamics programs by avoiding differential equation theory altogether and using optimization techniques to satisfy the system of nonlinear equations at each time step.

By enforcing a time invariant constraint on the system equations, accurate results were obtained for problems of static equilibrium and steady state motion. Static problems included rods in tension and torsion, beams in bending and shear, the cantilevered elastica with a tip moment, and a long flexible beam with a transverse follower force. Steady-state problems were solved using the same constraints and included steady rotation for several geometries, with and without transverse loading, such as would be experienced by a rotor blade. With problems where the analytical solution exists, convergence was demonstrated to within a percent of the exact solution. Other problems were solved such that the mean change in the displacement solution was less than one percent between the finest meshes. This initial algorithm demonstrated satisfactory performance in obtaining steady-state solutions to problems, which allows a full set of initial conditions to be obtained for any dynamic problem.

Testing the algorithm on steady-state problems was an iterative process, involving several changes to improve robustness for different classes of problems. It revealed some important lessons for solving multibody dynamics problems with this approach. An incremental technique was developed for nonlinear problems such that the initial guess is always close to the solution. Numerical derivatives were used to

project the guess for subsequent time steps in dynamic simulations, and unit changes were required to improve scaling in the state vector. Finally, the use of an analytical Jacobian improved efficiency and run times significantly.

Removing the time invariant constraint and substituting values for initial conditions allowed the algorithm to solve dynamic problems. The full set of nonlinear equations was solved at each time step, generating a new set of initial conditions for subsequent iterations. This approach was demonstrated for axial, torsional, and transverse free vibration in beams and for simple maneuvers such as periodic flapping and an angular spin up. Frequency response of the data revealed vibrations at the expected natural frequencies for all cases, while the time response during maneuvers behaved as expected. The exception occurred in free vibrations with a single element. For this case the peak frequency of vibration was sensitive to the time step, increasing or decreasing with the element aspect ratio, and matching the exact frequency only when the aspect ratio was equal to the speed of sound in the material. This phenomenon was not repeated with larger meshes, and its source is unknown at this time.

5.2 Recommendations for Future Research

The intention of this research is to serve as a proof of concept for the application of HWP to flexible body dynamics. While many classes of problems were investigated, the algorithm is designed to handle much more. For example, nearly all simulations were performed on prismatic, isotropic beams. This was done for simplicity, in order to test the code against common problems and obtain reasonable solutions. Further applications should successfully demonstrate the application to problems of initially curved and twisted beams, variable geometries, and orthotropic and anisotropic materials. The algorithm is fully equipped to handle these problems.

Every simulation was performed with a cantilevered boundary condition. This was also done for simplicity, as a fixed-free boundary condition is the easiest to model. The algorithm assembles the unknowns into a state vector of length $42N_x$. With

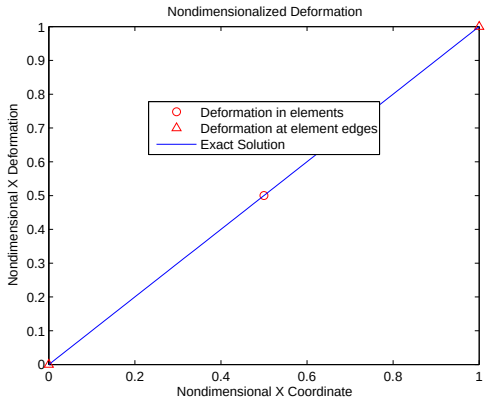
different boundary conditions, the unknowns change, requiring the state vector to be rearranged. This also changes the form of the Jacobian, which is $42N_x$ by $42N_x$ and ties directly to the ordering of the state vector. The cantilevered boundary condition is easy to model, because the force and moment vectors at the tip and displacement and rotation vectors at the root are known. With a pinned beam, the rotation and moment vectors are split between known and unknown quantities. The relationship between the rotation parameters must be prescribed for the planar motion of a pinned beam. Because these add complexity to a new algorithm, they were omitted for this research. An architecture that allows the incorporation of all types of boundary conditions must be developed for this algorithm to be useful.

The extension of this algorithm to a true *multibody* dynamics problem has not yet been demonstrated. A cantilevered beam is a single body, no matter how many elements it is discretized into. All elements are continuous, constraining their shared boundaries to the same values of force, moment, displacement, and rotation. This worked well for the current research, simplifying the assembly procedure. However, the extension to multibody dynamics will require the ability to handle discontinuities at the element boundaries in order to connect elements with various types of joints. For example, two beam elements are connected by a pinned joint if their shared edges have equal values for displacement and force but not rotation and moment. Each element has 24 scalar quantities assigned at its spatial boundaries (from $\hat{F}_1$, $\hat{F}_2$, $\hat{M}_1$, $\hat{M}_2$, $\hat{q}_4$, $\hat{q}_2$, $\hat{\rho}_4$, and $\hat{\rho}_2$). Of these quantities, 12 must be specified as boundary conditions or tied to another element by a specified joint. A more generalized architecture will make use of this requirement during the assembly process.

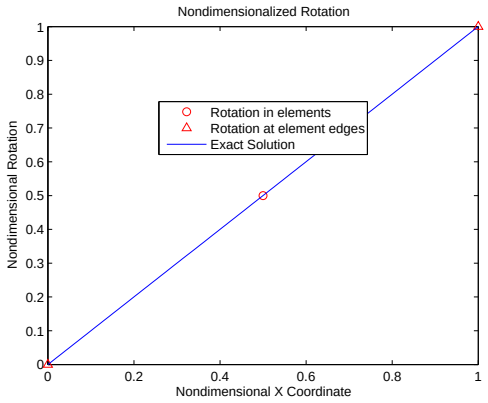
A disadvantage to the current approach is that complex problems require some knowledge of the solution in advance in order to appropriately scale the state vector. A robust solution technique will provide a generalized method for scaling, possibly by nondimensionalizing all quantities in the derivation. This has not been fully explored and may provide a way to eliminate experimenting with units in order to get the solution to converge.

The appearance of high frequency content in the time accurate solutions was significant (particularly in the 3-D solutions), though the fundamental frequencies of vibration were correct. Numerical round off errors can add high frequency noise and transfer energy to the lower frequencies, as pointed out by Bauchau [11]. This effect could be reduced by the implementation of an energy dissipation algorithm. Inclusion of an energy decay statement over each time interval will incorporate this method in the current algorithm, however at this time energy decay algorithms are only used with displacement formulations. It may become necessary to develop a finite element model which uses higher order shape functions in order to accomplish this. During the course of this research the use of a damping term was investigated, which generated forces that oppose the strain velocity of each element and dissipate energy. However, this was not fully developed and no results were presented.

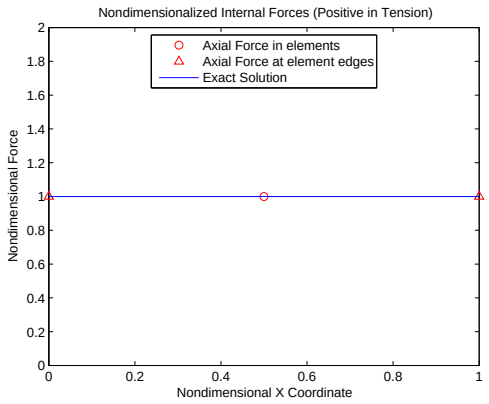
In addition to the suggested improvements for the current algorithm, extending the principles used to other types of elements (shells, plates, membranes, etc.) is necessary to investigate the full application of HWP in elastodynamics. This will require a separate development using the unique constitutive, kinematic, and energy relationships for these types of bodies. An architecture for a software code using this application will require an element library and an efficient way to assemble the state vector that involves the more complex boundary conditions associated with connections between different element types. The research presented serves as a proof of concept using a beam element, and is a first step in this direction.



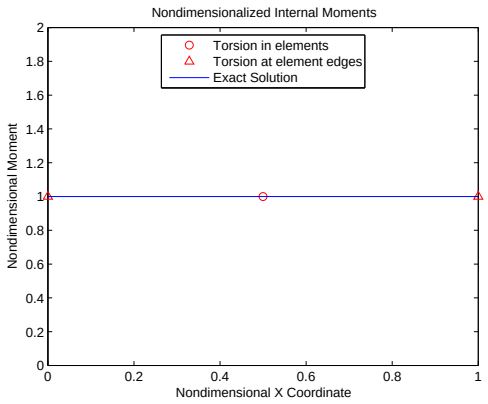
(a) Deformation



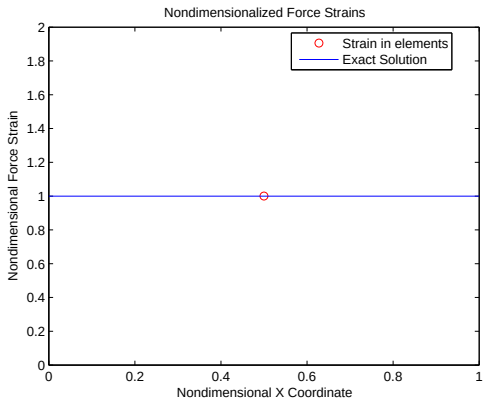
(b) Rotation



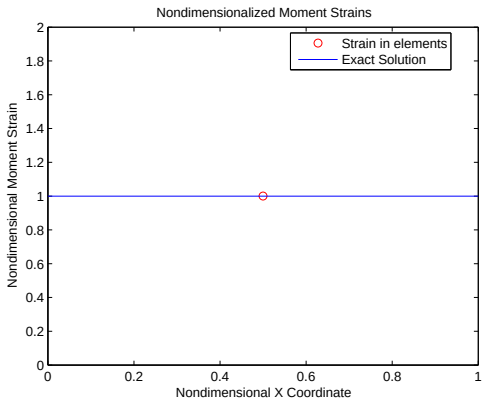
(c) Internal Force



(d) Internal Moment

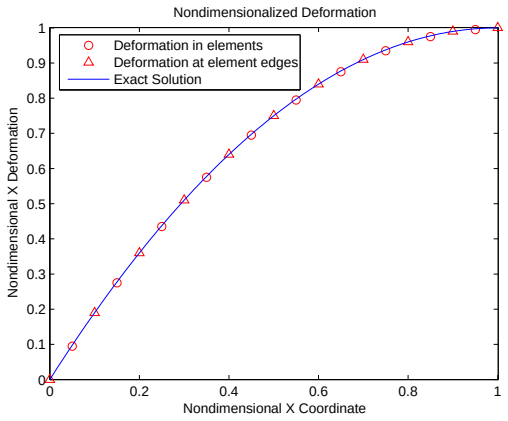


(e) Force Strain

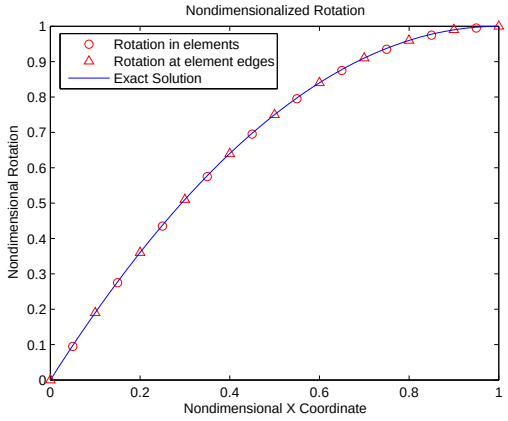


(f) Moment Strain

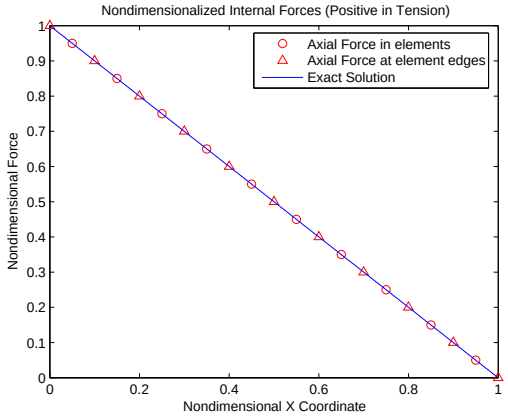
Figure A.1: Nondimensionalized solution for a fixed-free rod with an axial tip load and torque.



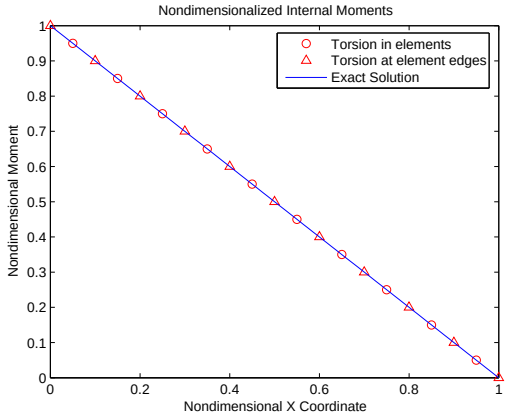
(a) Deformation



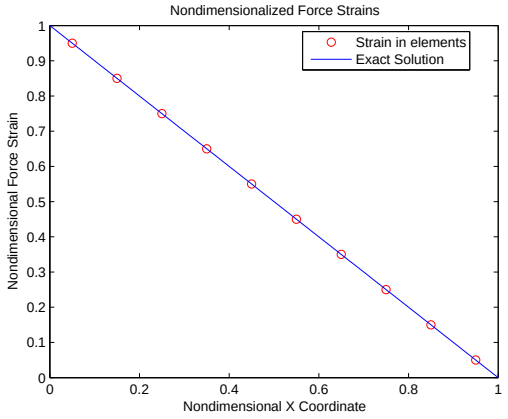
(b) Rotation



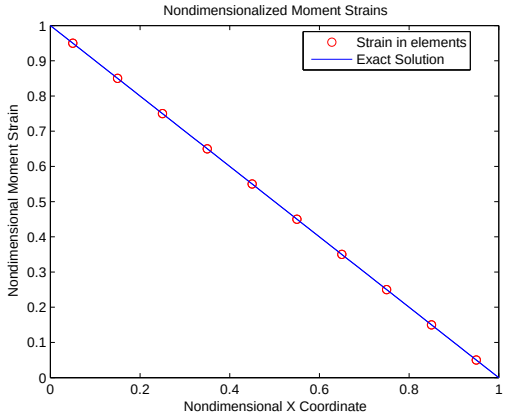
(c) Internal Force



(d) Internal Resultant

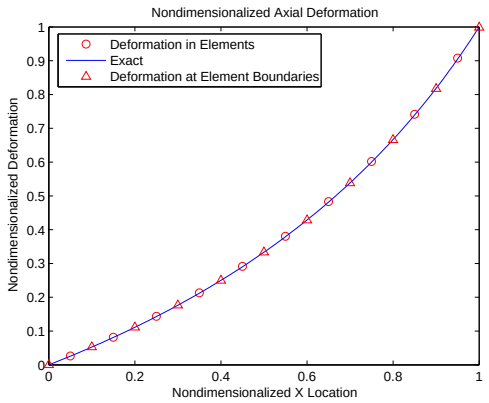


(e) Force Strain

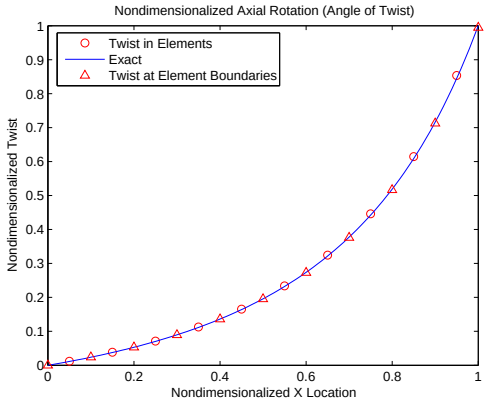


(f) Moment Strain

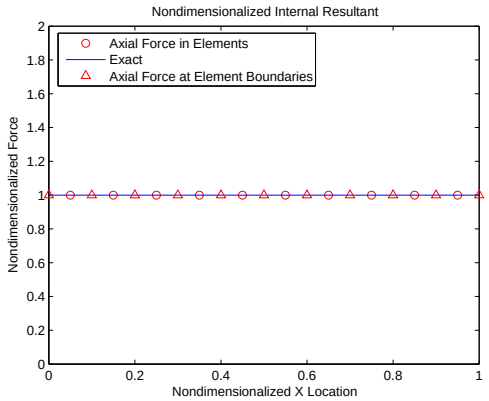
Figure A.2: Nondimensionalized solution for a fixed-free rod with a distributed axial load and torque.



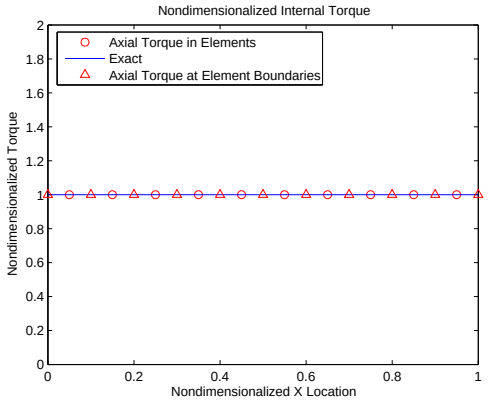
(a) Deformation



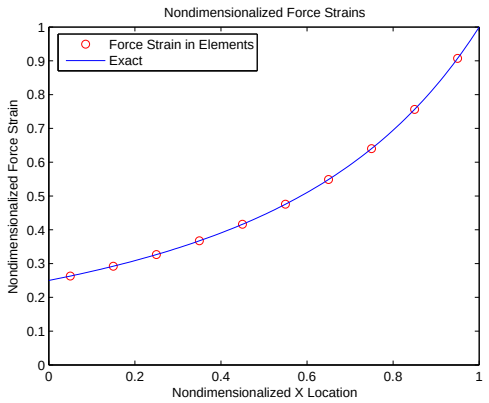
(b) Twist



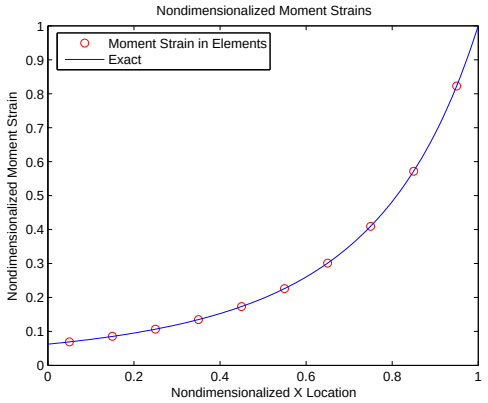
(c) Internal Force



(d) Internal Torque

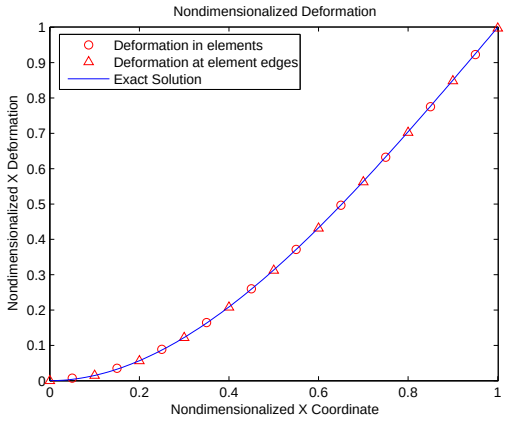


(e) Force Strain

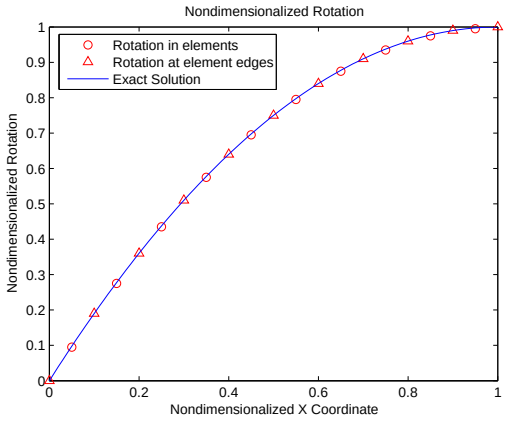


(f) Moment Strain

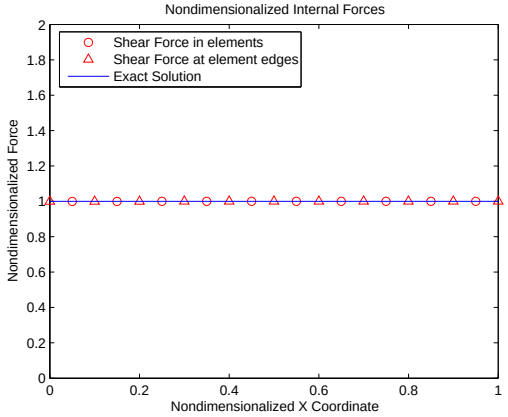
Figure A.3: Nondimensionalized solution for a fixed-free tapered rod with a tip load and torque.



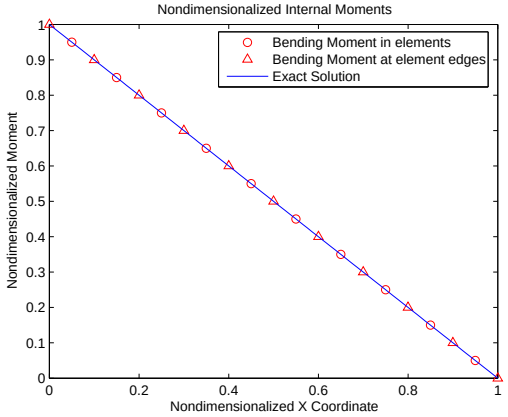
(a) Deformation



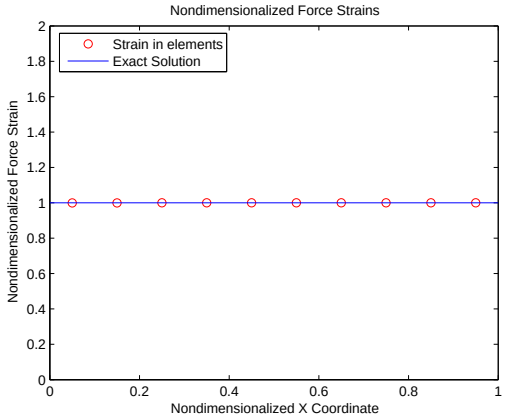
(b) Rotation



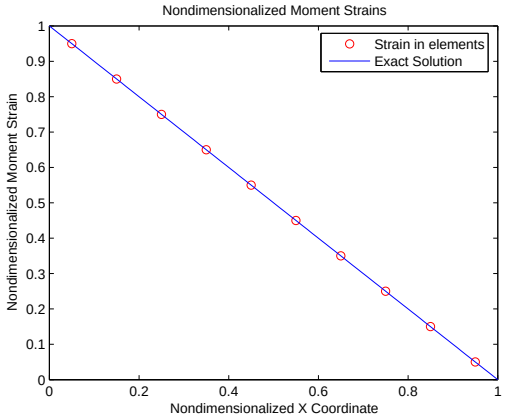
(c) Internal Force



(d) Internal Moment

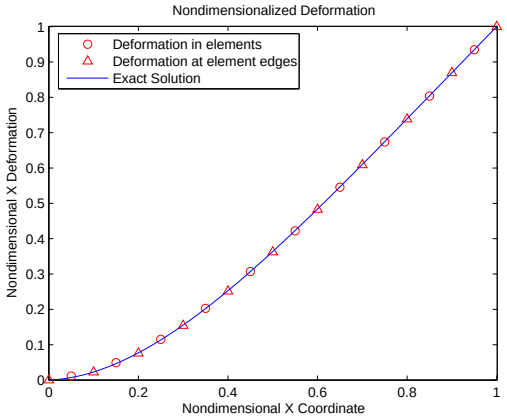


(e) Force Strain

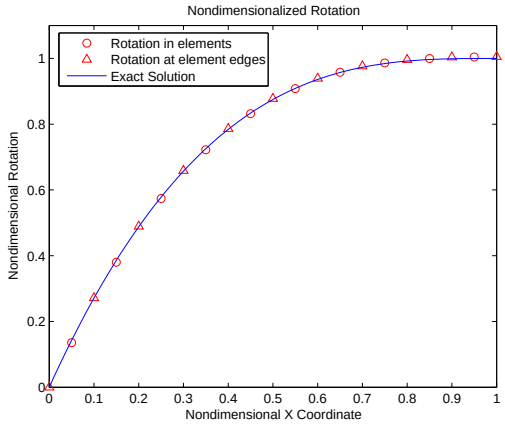


(f) Moment Strain

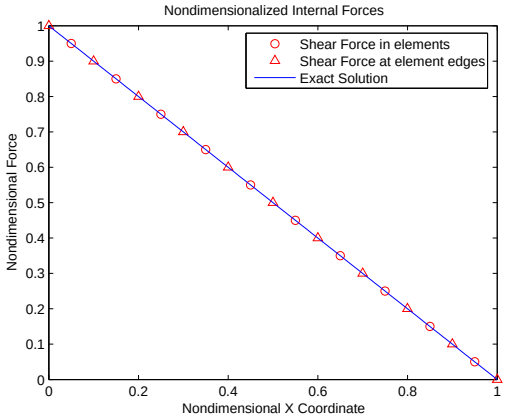
Figure A.4: Nondimensionalized solution for a cantilevered beam with a transverse tip load.



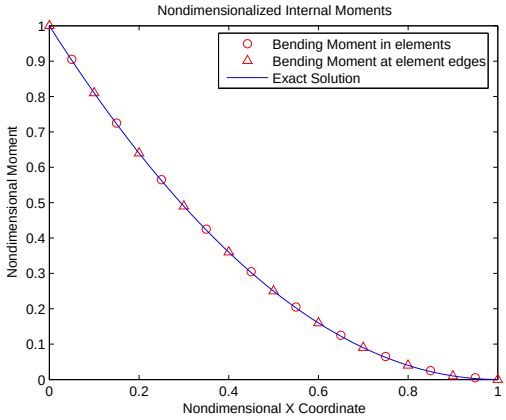
(a) Deformation



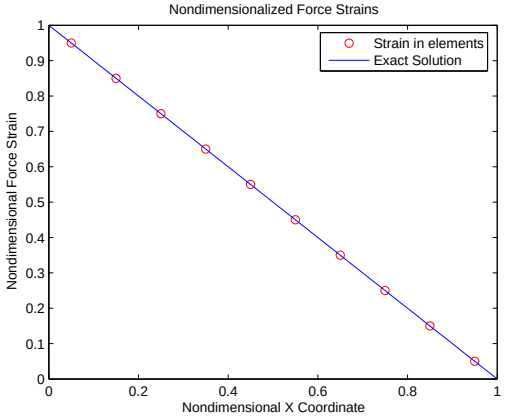
(b) Rotation



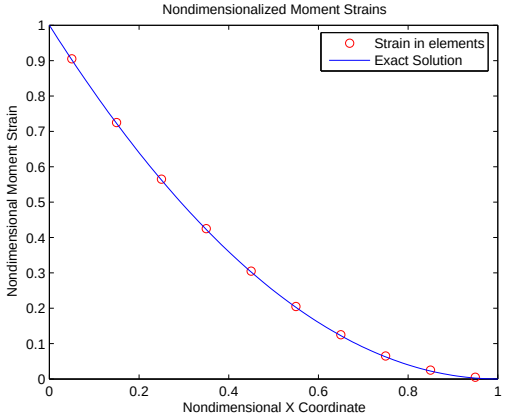
(c) Internal Force



(d) Internal Moment

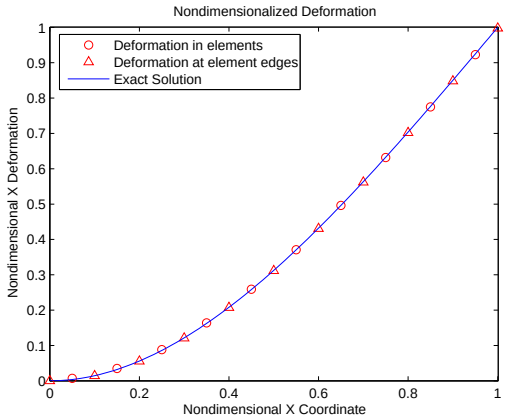


(e) Force Strain

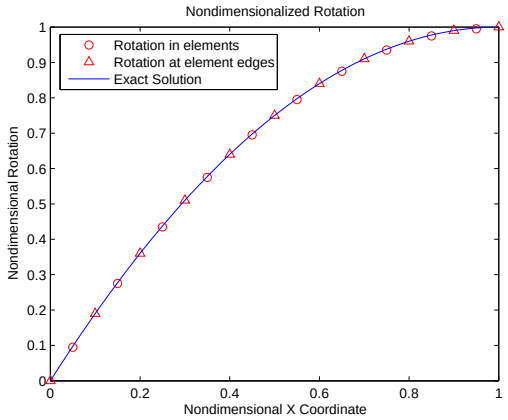


(f) Moment Strain

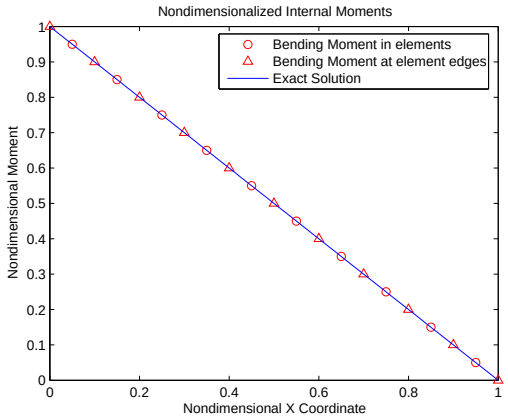
Figure A.5: Nondimensionalized solution for a cantilevered beam with a distributed transverse load.



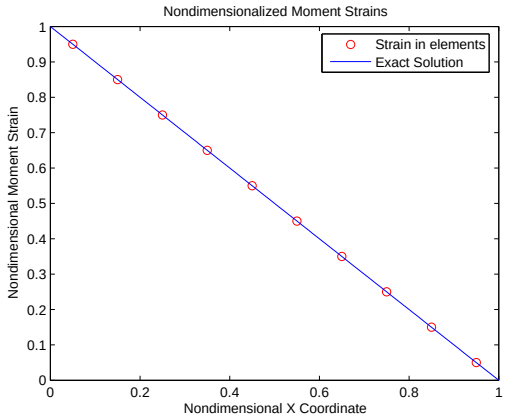
(a) Deformation



(b) Rotation

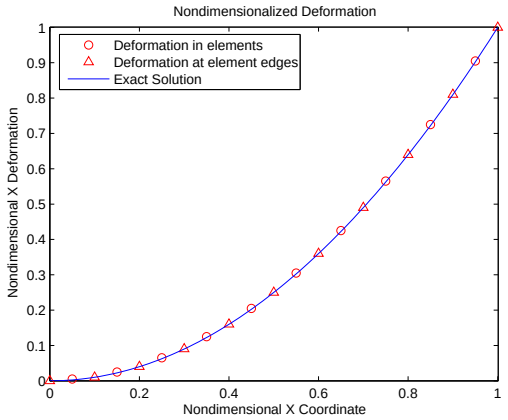


(c) Internal Moment

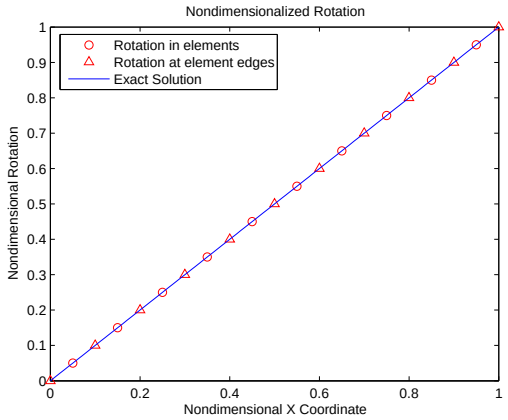


(d) Moment Strain

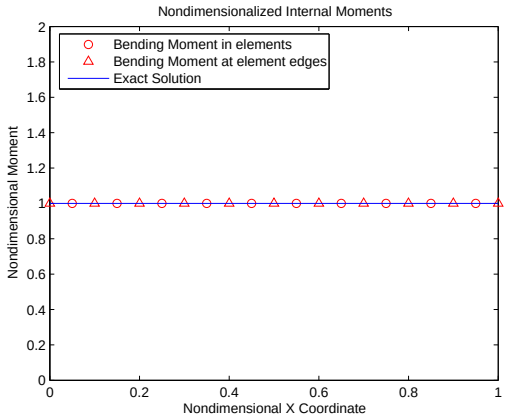
Figure A.6: Nondimensionalized solution for a cantilevered beam with a distributed bending moment.



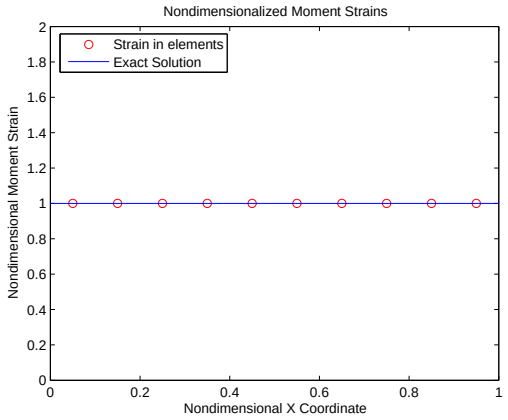
(a) Deformation



(b) Rotation

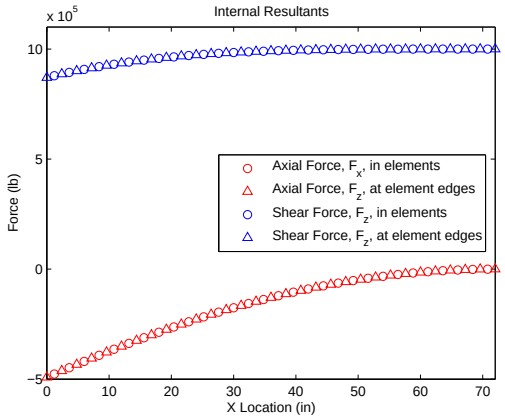


(c) Internal Moment

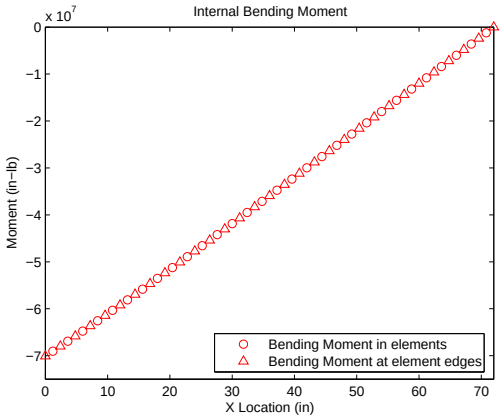


(d) Moment Strain

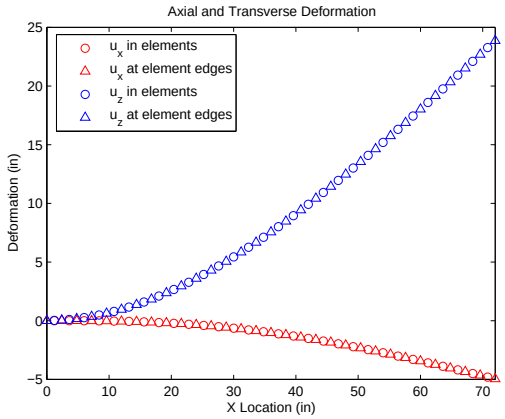
Figure A.7: Nondimensionalized solution for a cantilevered beam with a bending moment applied at the tip.



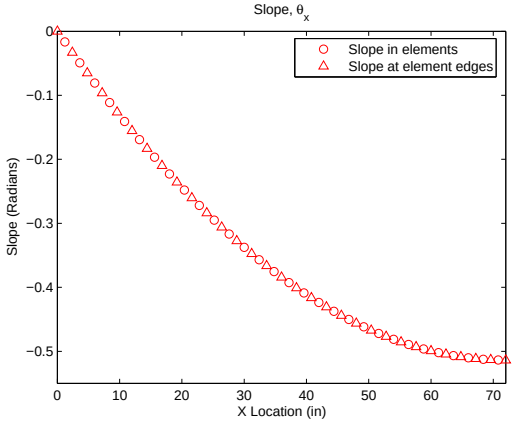
(a) Internal Force



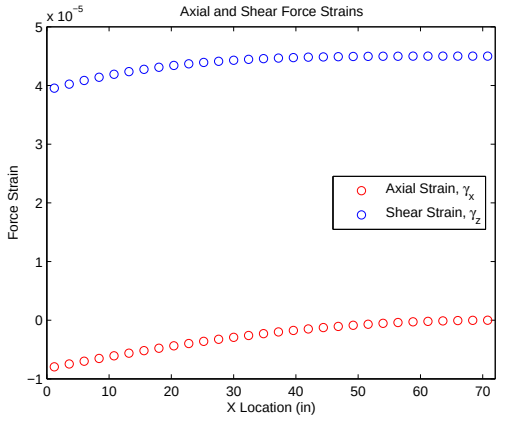
(b) Internal Moment



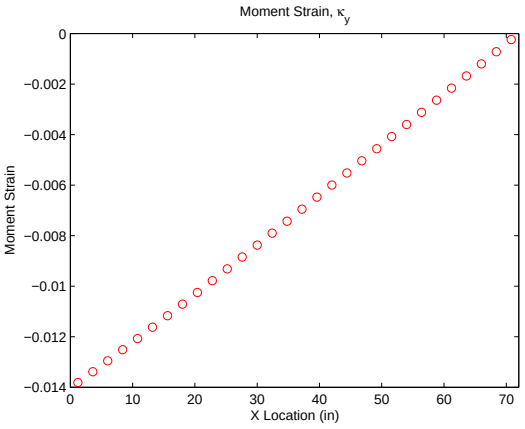
(c) Deformation



(d) Rotation

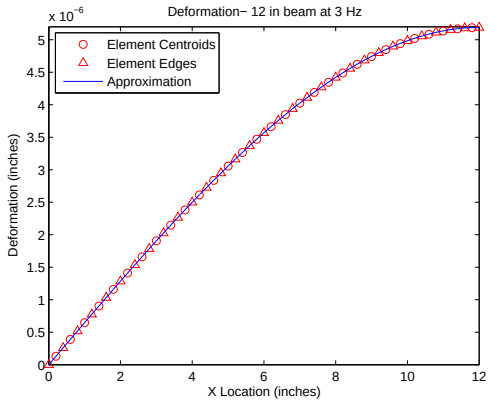


(e) Force Strain

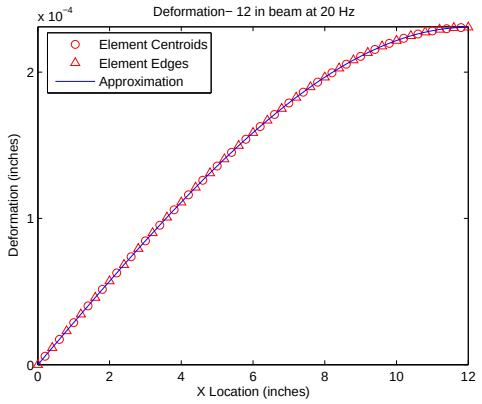


(f) Moment Strain

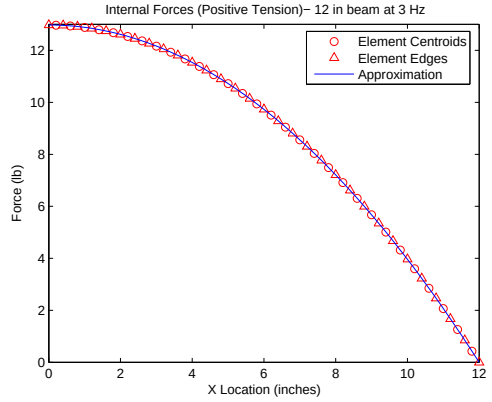
Figure A.8: Converged solution for a 72 inch cantilevered aluminum beam with a 1000 lb transverse follower force.



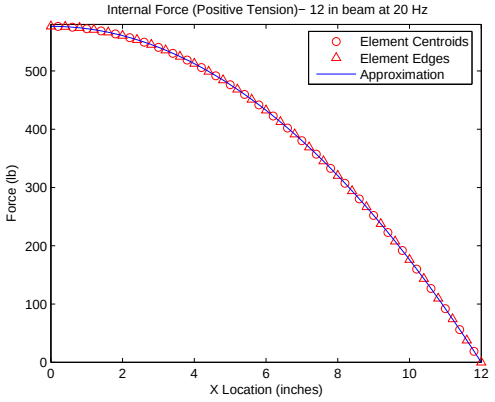
(a) Deformation at 3 Hz



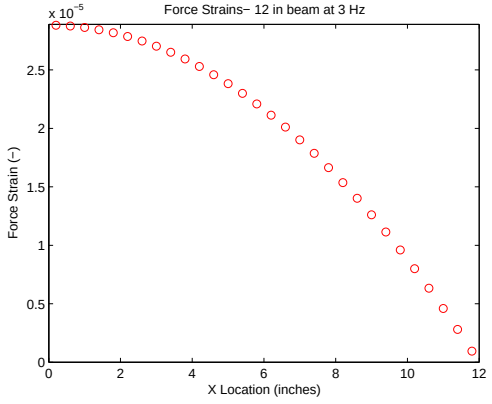
(b) Deformation at 20 Hz



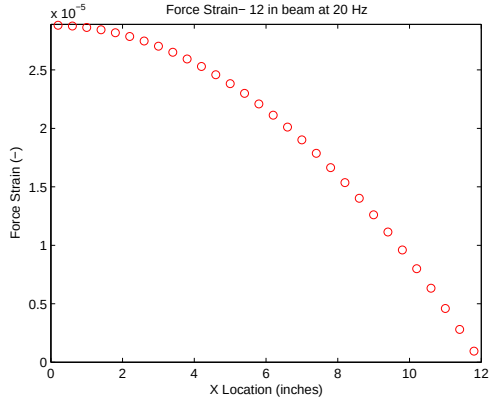
(c) Internal Force at 3 Hz



(d) Internal Force at 3 Hz

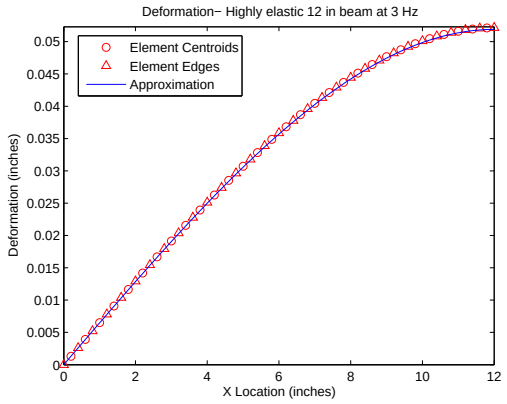


(e) Force Strain at 3 Hz

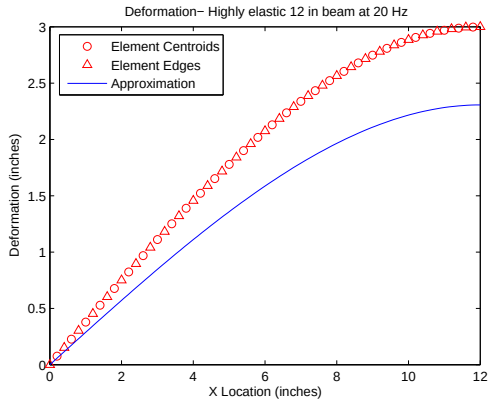


(f) Force Strain at 20 Hz

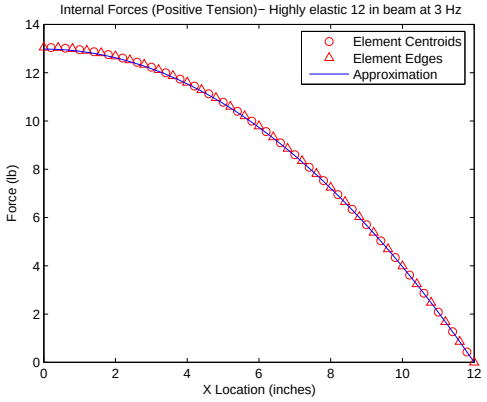
Figure A.9: Results for a cantilevered, 12 inch aluminum beam rotating at 3Hz and 20Hz about its vertical axis.



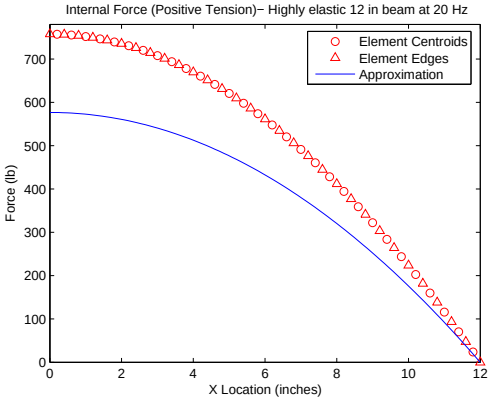
(a) Deformation at 3 Hz



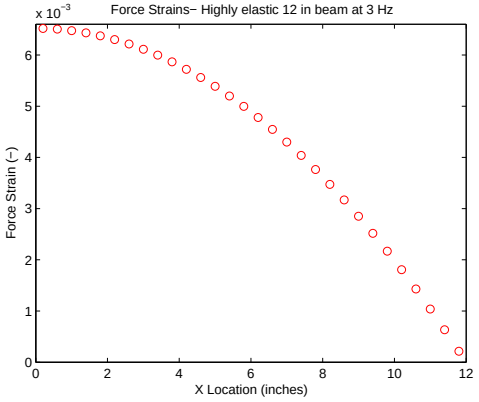
(b) Deformation at 20 Hz



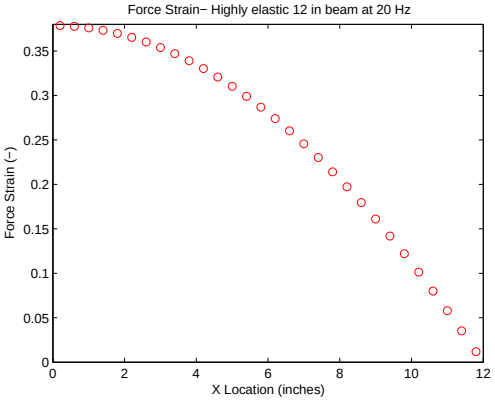
(c) Internal Force at 3 Hz



(d) Internal Force at 20 Hz

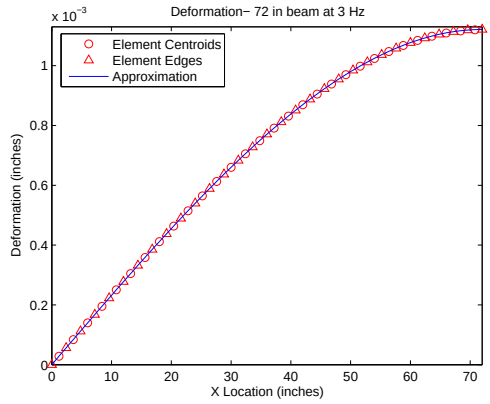


(e) Force Strain at 3 Hz

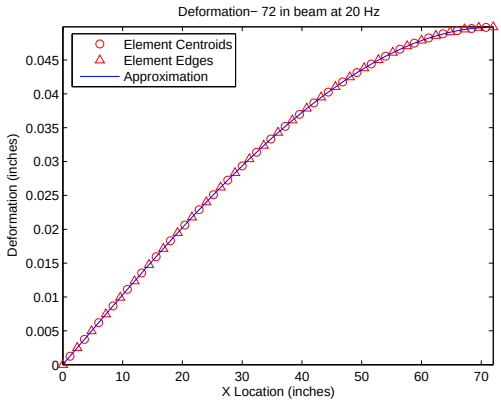


(f) Force Strain at 20 Hz

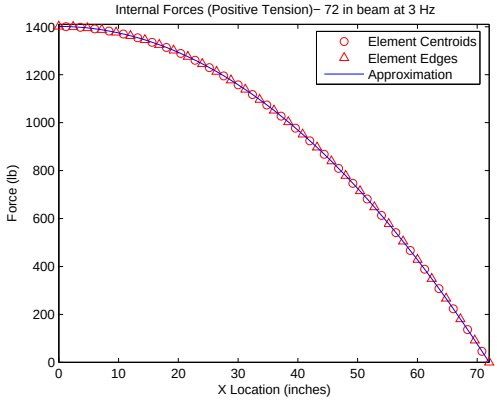
Figure A.10: Results for a cantilevered, 12 inch elastic beam rotating at 3Hz and 20Hz about its vertical axis.



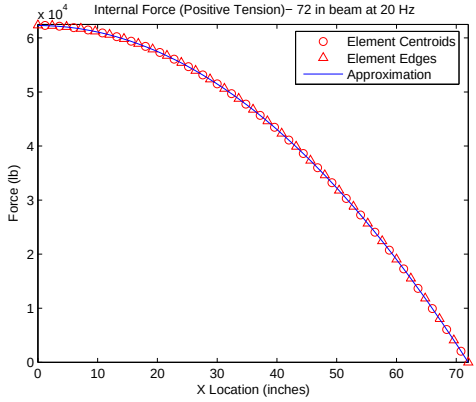
(a) Deformation at 3 Hz



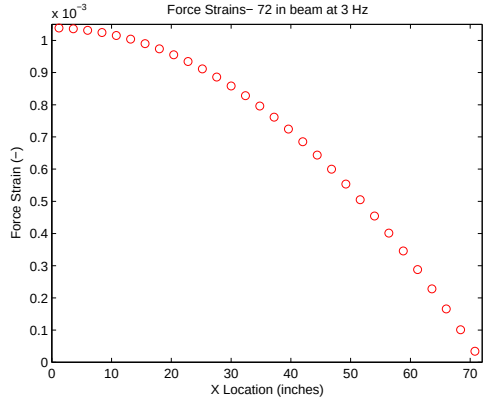
(b) Deformation at 20 Hz



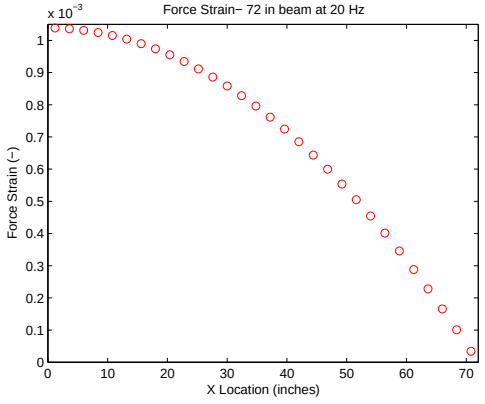
(c) Internal Force at 3 Hz



(d) Internal Force at 20 Hz

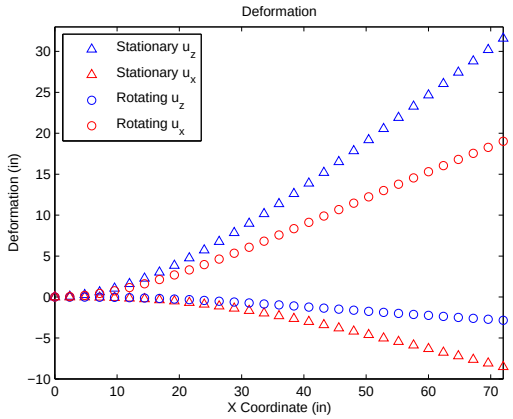


(e) Force Strain at 3 Hz

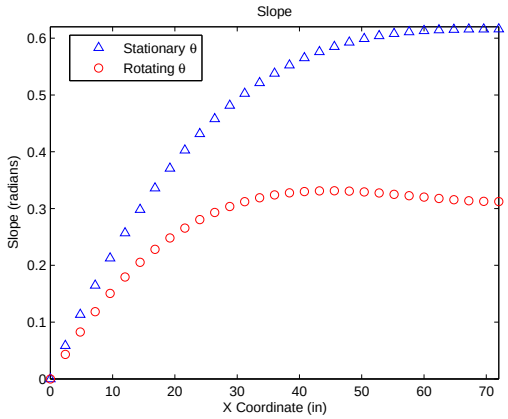


(f) Force Strain at 20 Hz

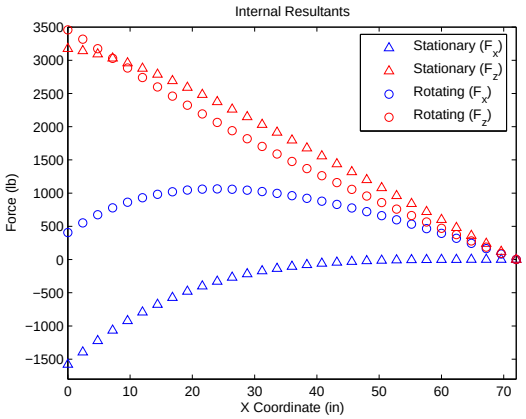
Figure A.11: Results for a cantilevered, 72 inch aluminum beam rotating at 3Hz and 20Hz about its vertical axis.



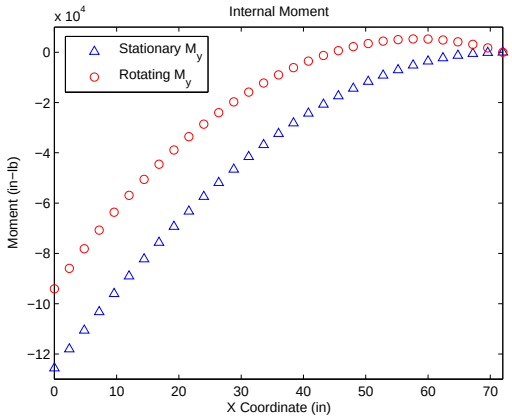
(a) Deformation



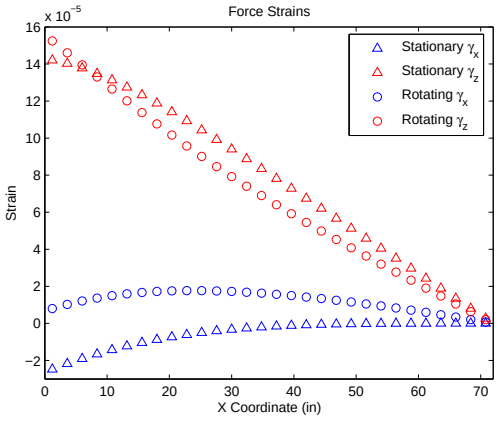
(b) Rotation



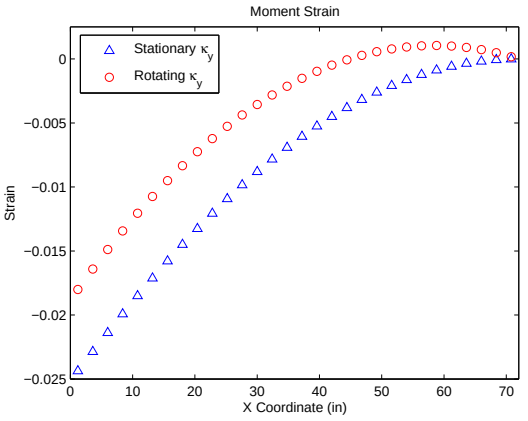
(c) Internal Force



(d) Internal Moment

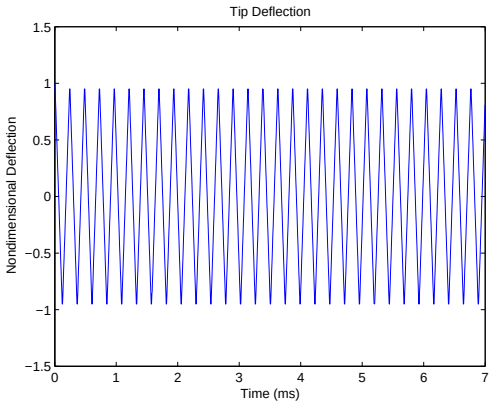


(e) Force Strain

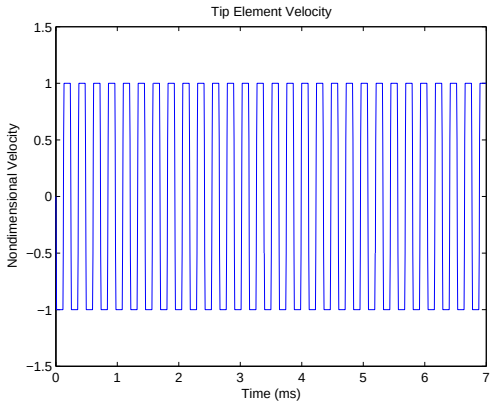


(f) Moment Strain

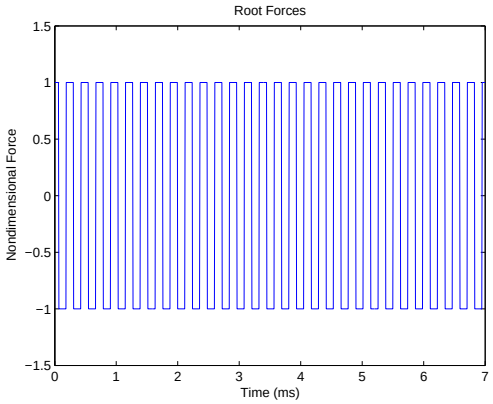
Figure A.12: Results for a cantilevered, 72 inch aluminum beam with a distributed transverse load under stationary and rotating conditions.



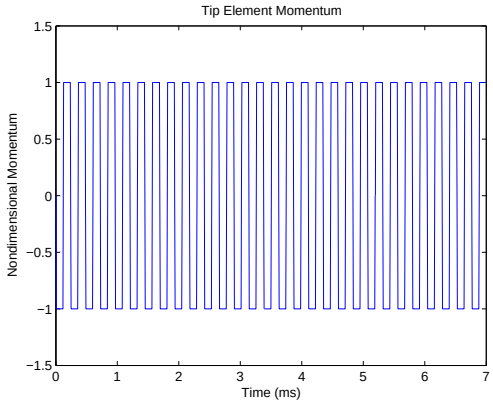
(a) Deflection



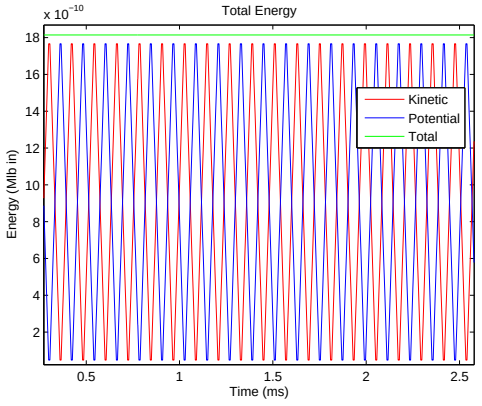
(b) Velocity



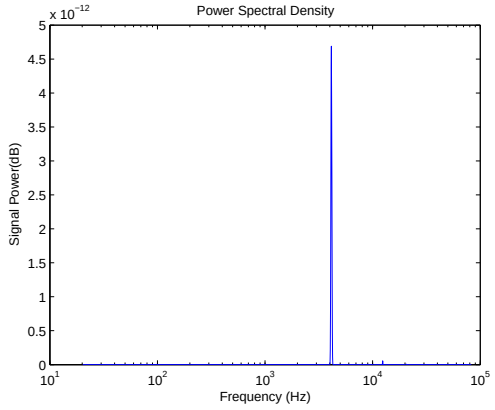
(c) Root Force



(d) Momentum

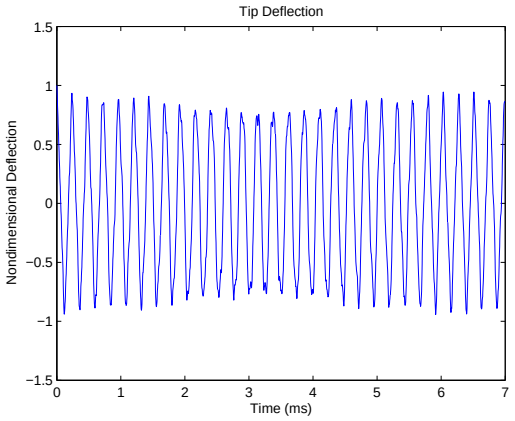


(e) Energy

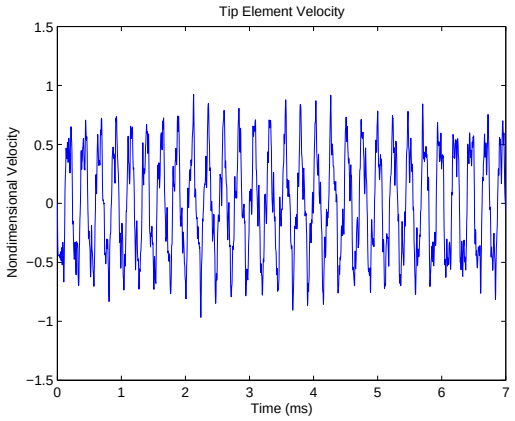


(f) Frequency Response

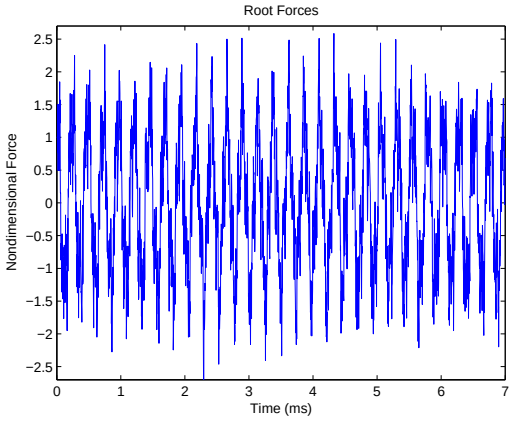
Figure B.1: Nondimensionalized axial free vibration of a fixed-free rod, $\Delta x/\Delta t = \sqrt{E/\rho}$, 10 elements, $\Delta t = 0.006ms$.



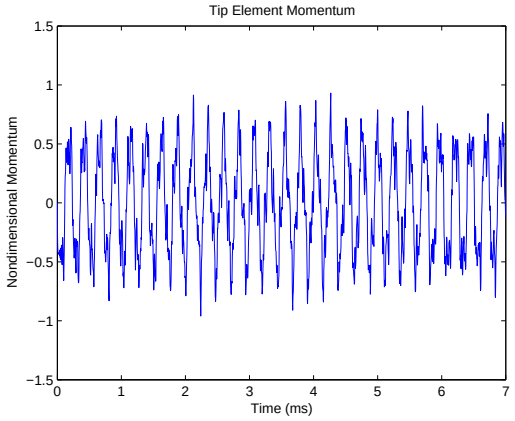
(a) Deflection



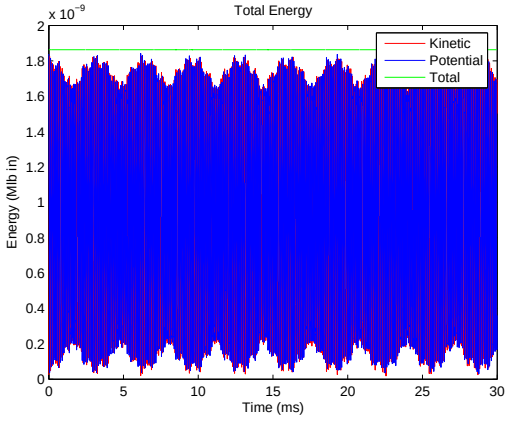
(b) Velocity



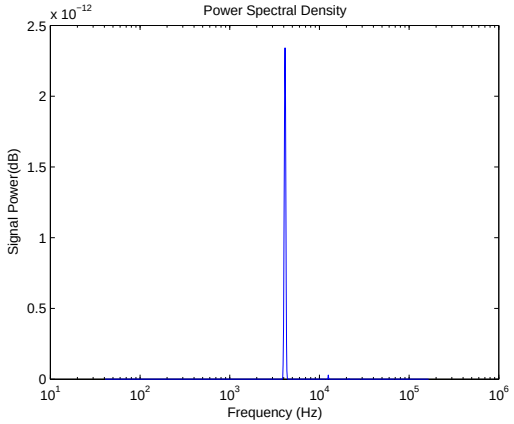
(c) Root Force



(d) Momentum

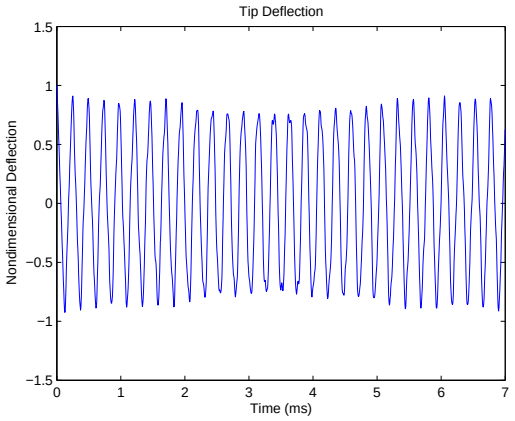


(e) Energy

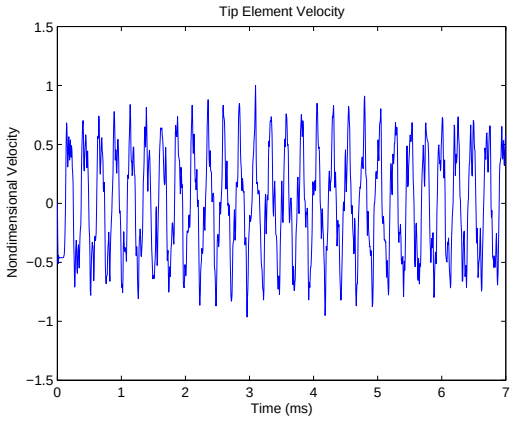


(f) Frequency Response

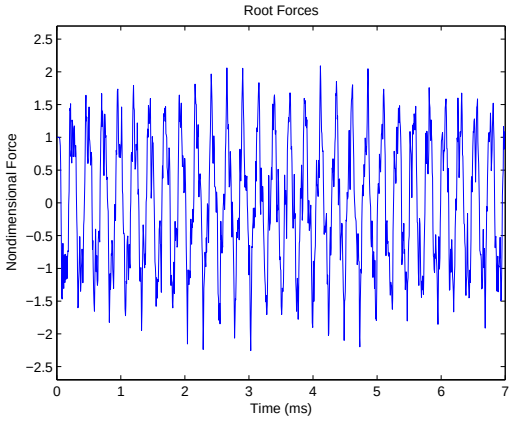
Figure B.2: Nondimensionalized axial free vibration of a fixed-free rod, $\Delta x/\Delta t \approx 2\sqrt{E/\rho}$, 10 elements, $\Delta t = 0.003ms$



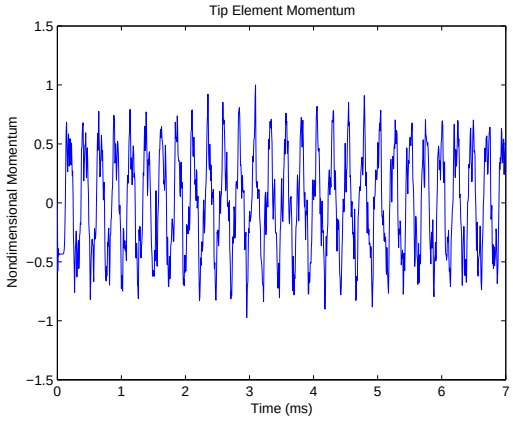
(a) Deflection



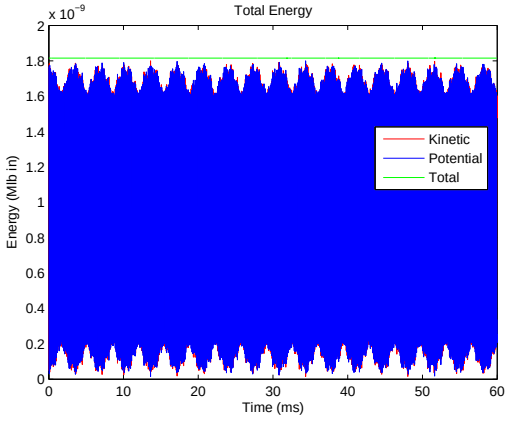
(b) Velocity



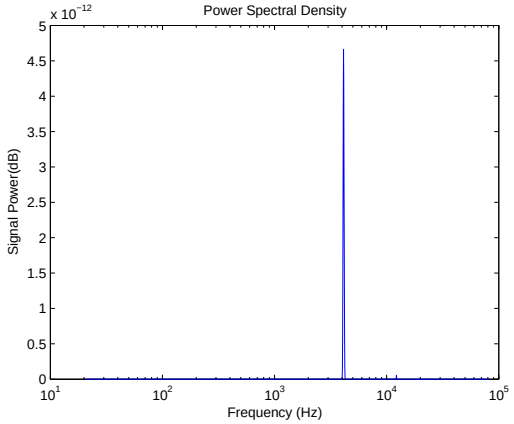
(c) Root Force



(d) Momentum

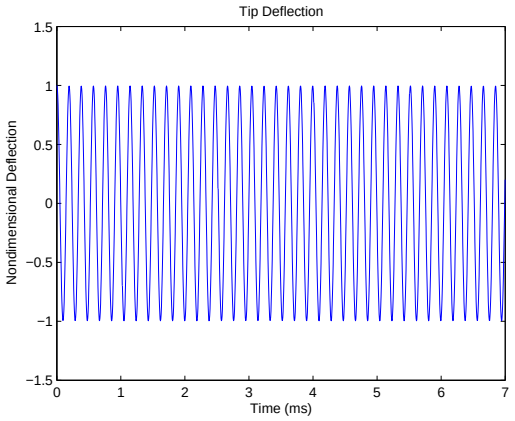


(e) Energy

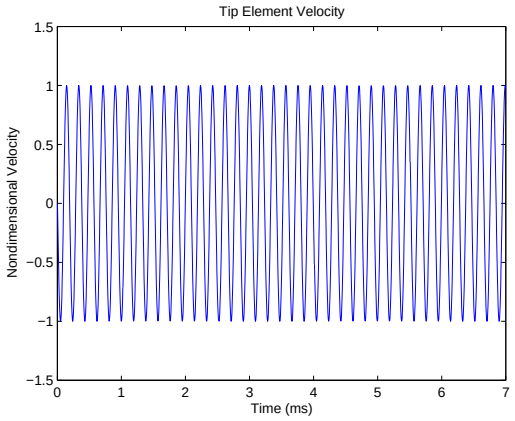


(f) Frequency Response

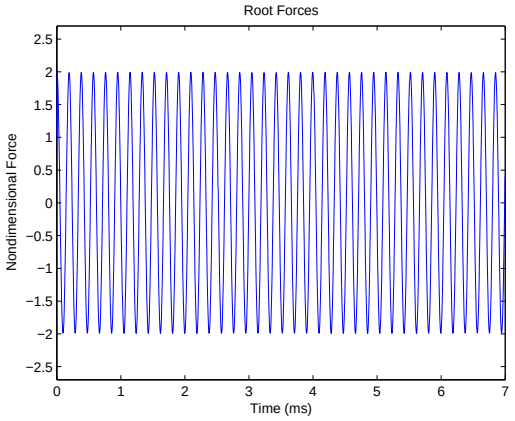
Figure B.3: Nondimensionalized axial free vibration of a fixed-free rod, $\Delta x/\Delta t \approx 0.5\sqrt{E/\rho}$, 20 elements, $\Delta t = 0.006ms$



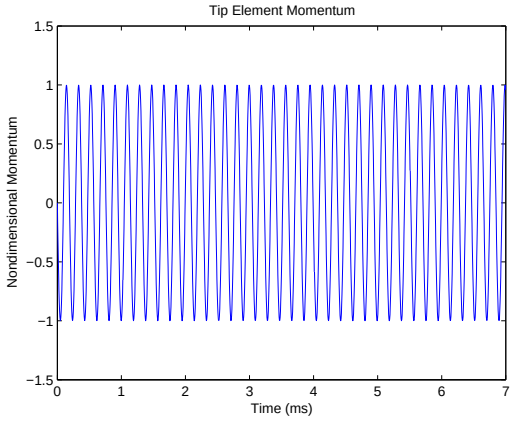
(a) Deflection



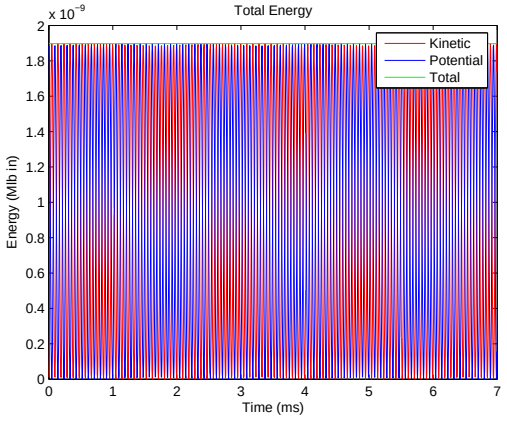
(b) Velocity



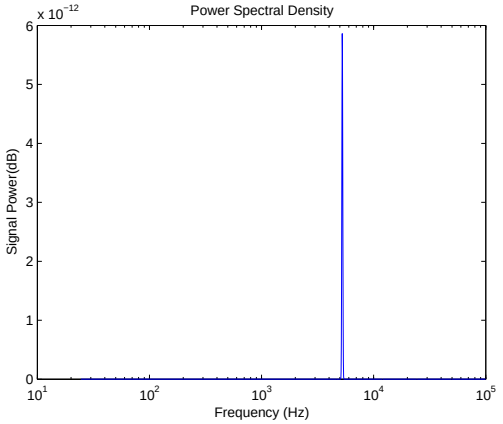
(c) Root Force



(d) Momentum

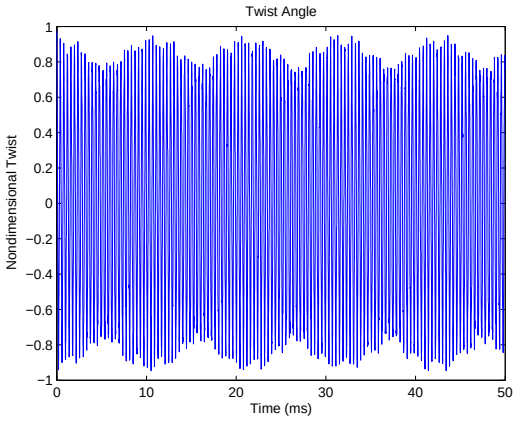


(e) Energy

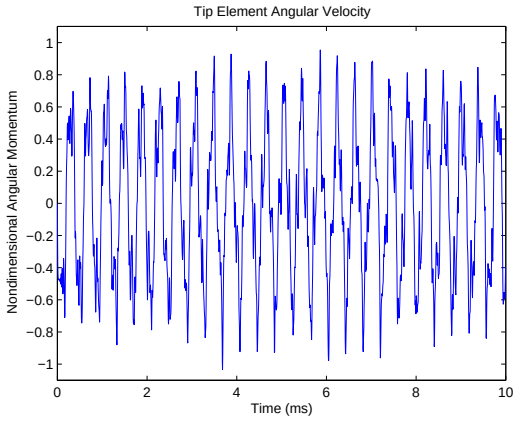


(f) Frequency Response

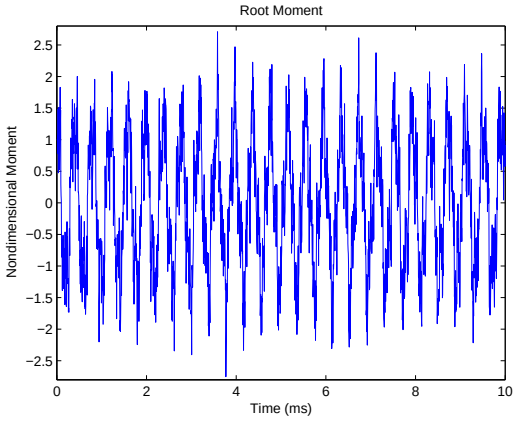
Figure B.4: Nondimensionalized axial free vibration of a fixed-free rod, $\Delta x/\Delta t \approx 12\sqrt{E/\rho}$, 1 element, $\Delta t = 0.005ms$



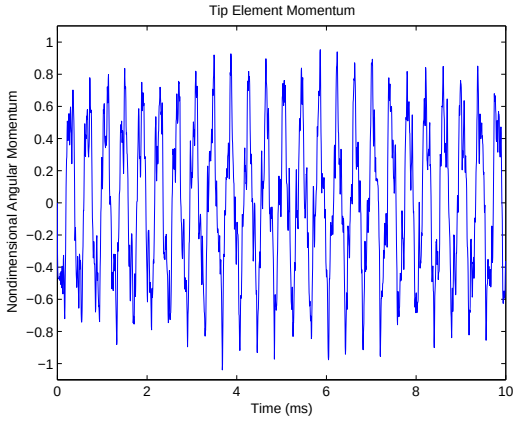
(a) Twist



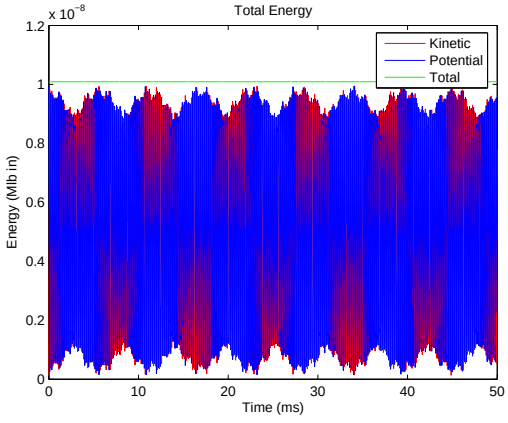
(b) Angular Velocity



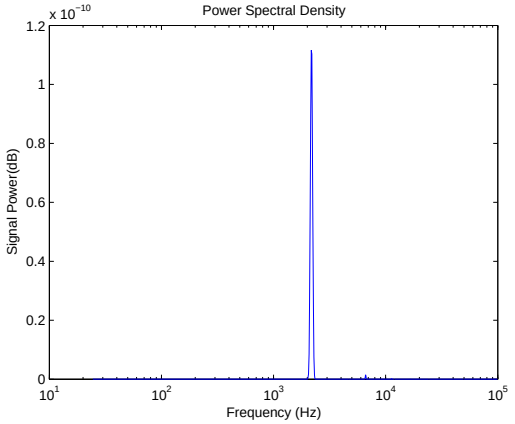
(c) Root Moment



(d) Angular Momentum

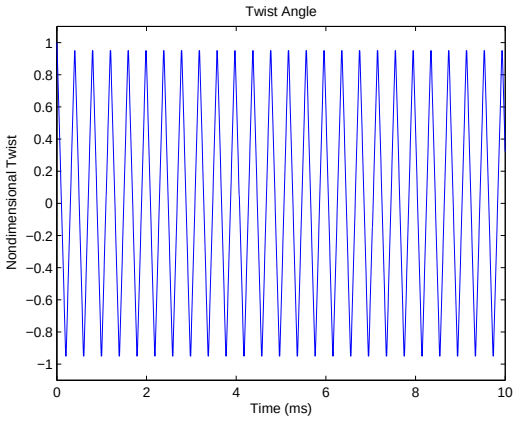


(e) Energy

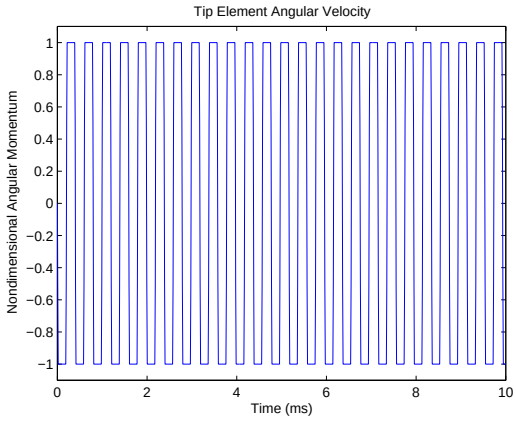


(f) Frequency Response

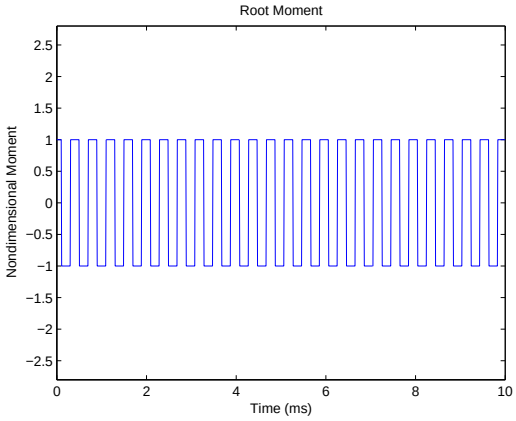
Figure B.5: Nondimensionalized torsional free vibration of a fixed-free rod, $\Delta x/\Delta t \approx 2\sqrt{E/\rho}$, 10 elements, $\Delta t = 0.005ms$



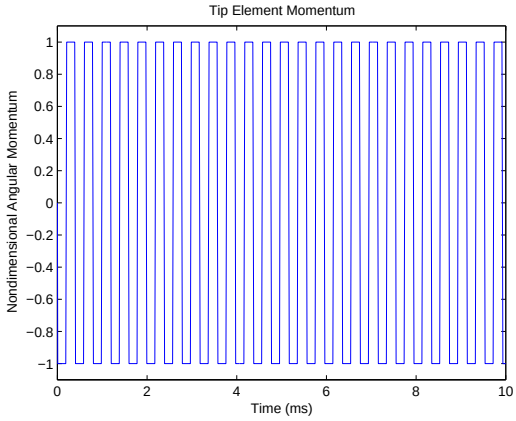
(a) Twist



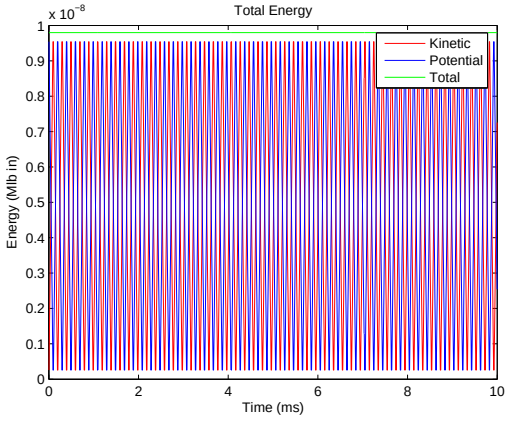
(b) Angular Velocity



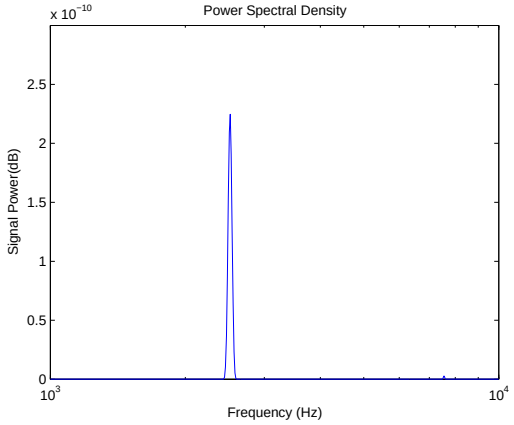
(c) Root Moment



(d) Angular Momentum

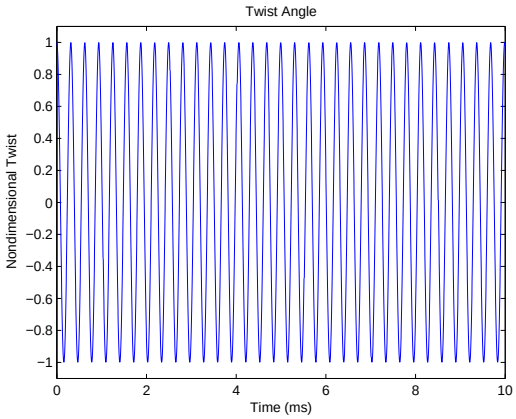


(e) Energy

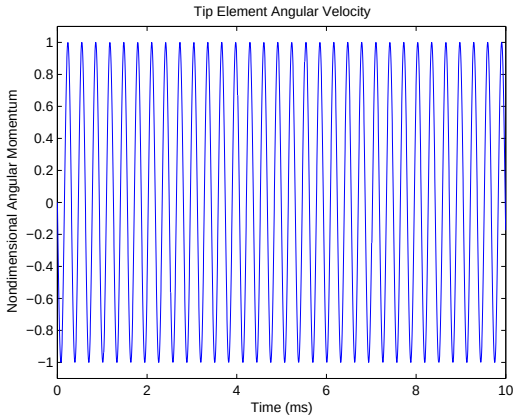


(f) Frequency Response

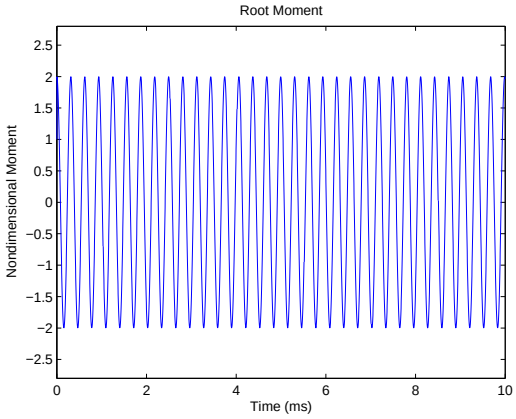
Figure B.6: Nondimensionalized torsional free vibration of a fixed-free rod, $\Delta x/\Delta t \approx \sqrt{G/\rho}$, 10 elements, $\Delta t = 0.0099302ms$



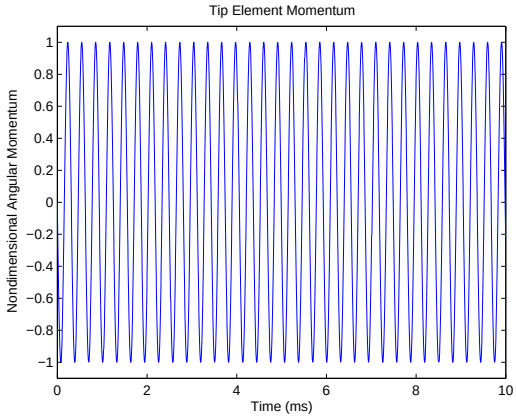
(a) Twist



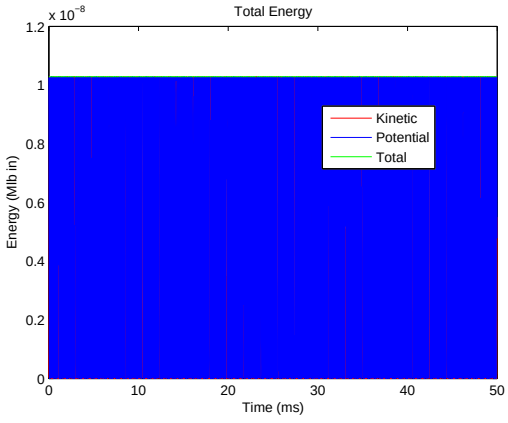
(b) Angular Velocity



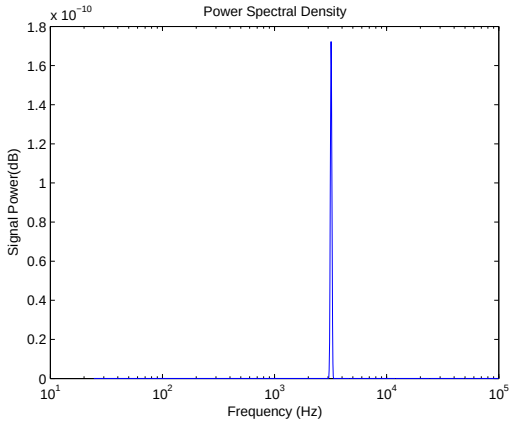
(c) Root Moment



(d) Angular Momentum

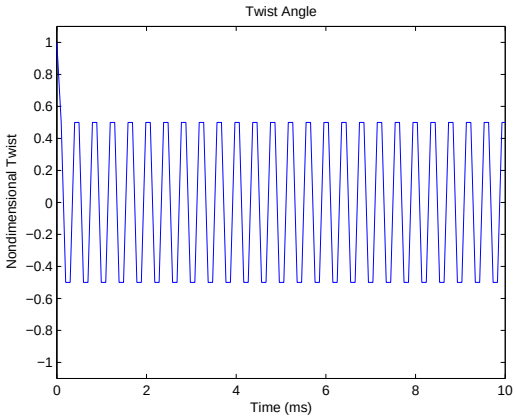


(e) Energy

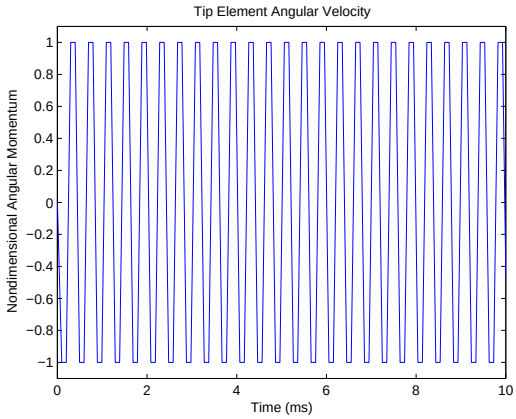


(f) Frequency Response

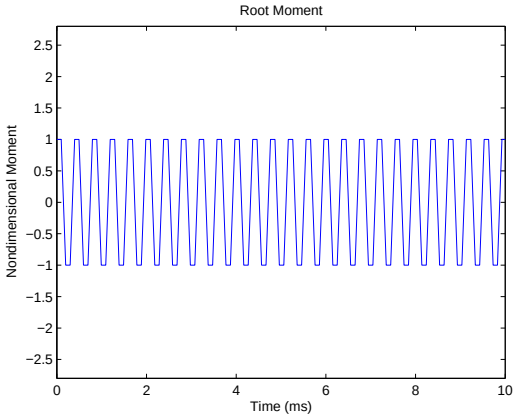
Figure B.7: Nondimensionalized torsional free vibration of a fixed-free rod, $\Delta x/\Delta t \approx 20\sqrt{G/\rho}$, 1 element, $\Delta t = 0.005ms$



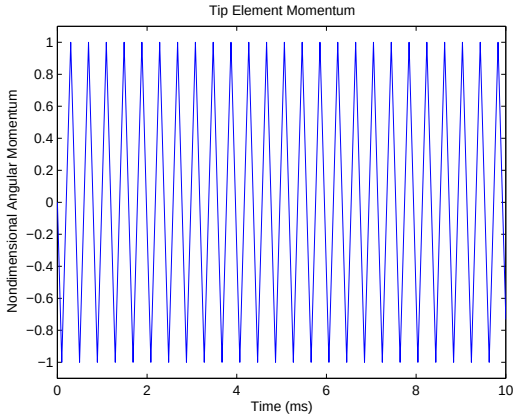
(a) Twist



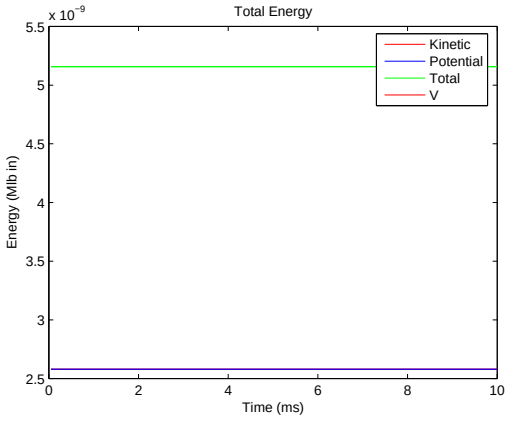
(b) Angular Velocity



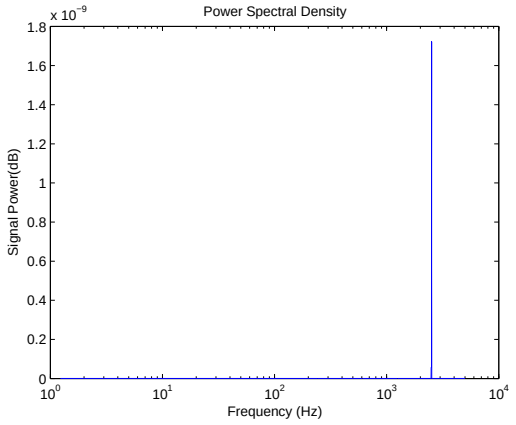
(c) Root Moment



(d) Angular Momentum

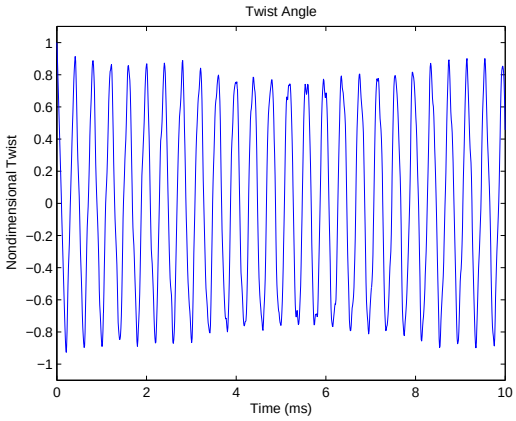


(e) Energy

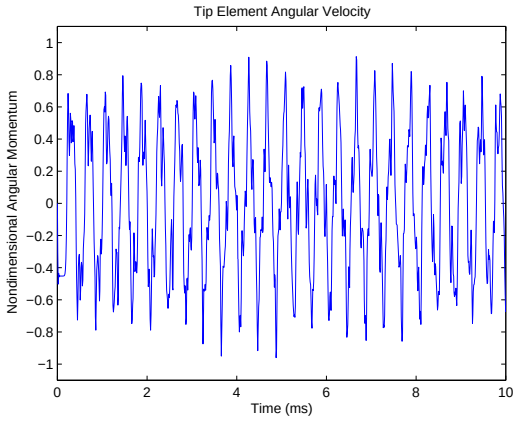


(f) Frequency Response

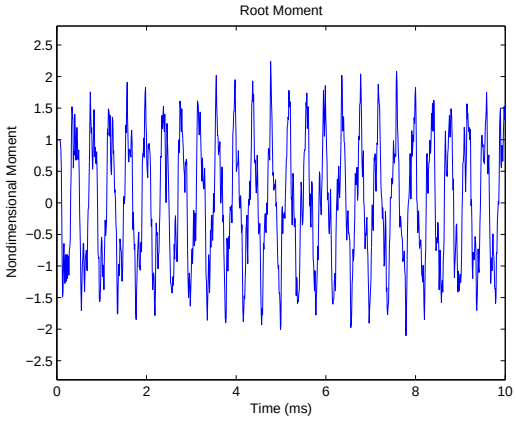
Figure B.8: Nondimensionalized torsional free vibration of a fixed-free rod, $\Delta x/\Delta t \approx \sqrt{G/\rho}$, 1 element, $\Delta t = 0.099302ms$



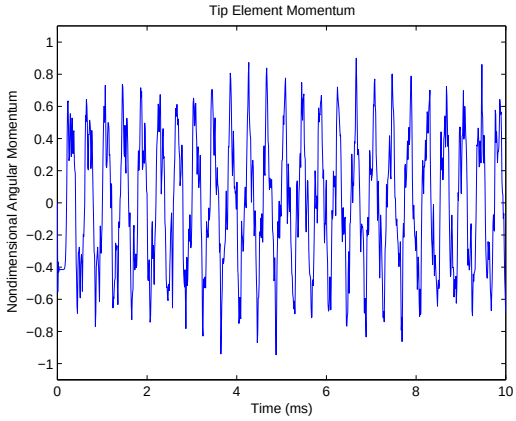
(a) Twist



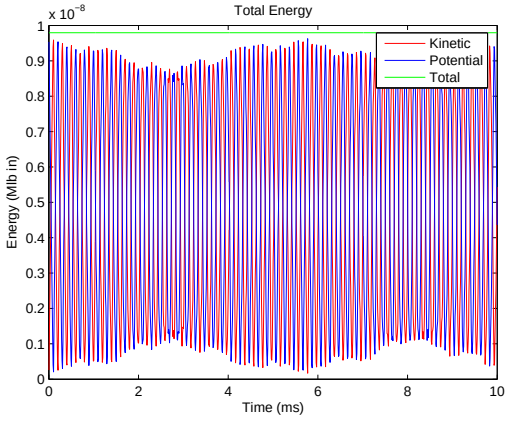
(b) Angular Velocity



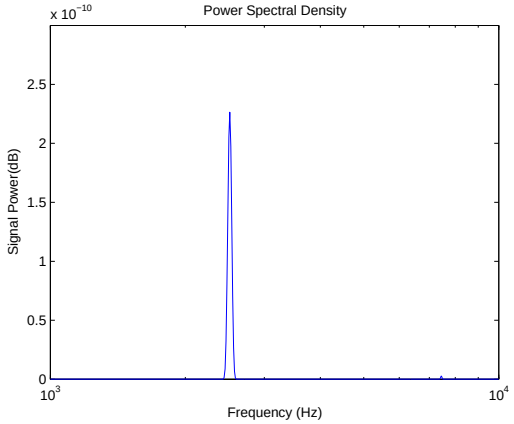
(c) Root Moment



(d) Angular Momentum

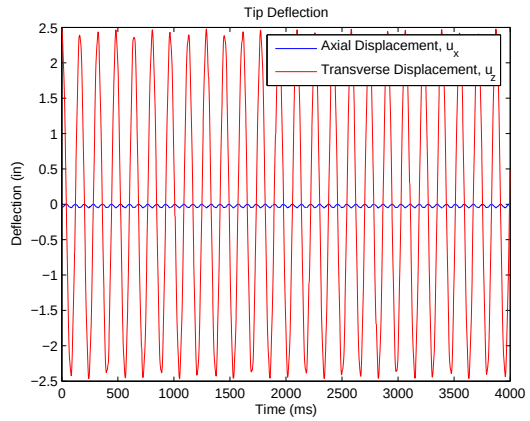


(e) Energy

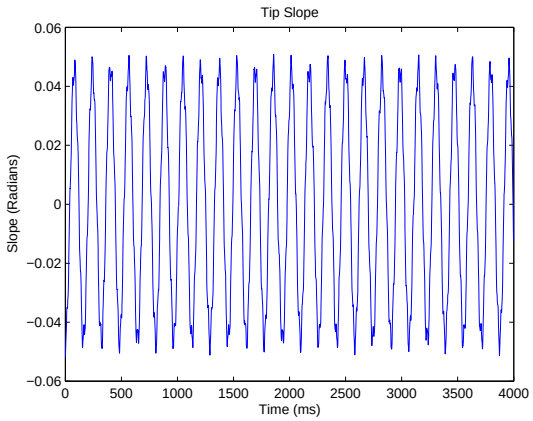


(f) Frequency Response

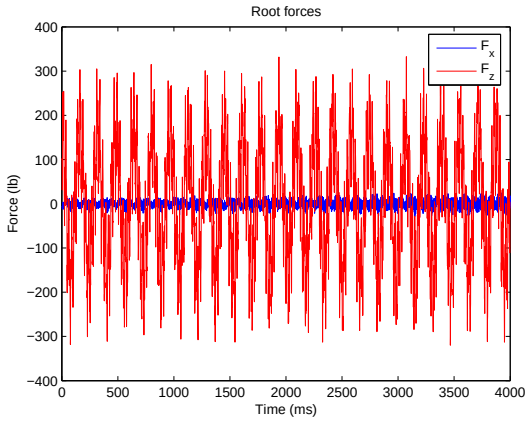
Figure B.9: Nondimensionalized torsional free vibration of a fixed-free rod, $\Delta x/\Delta t \approx 0.5\sqrt{G/\rho}$, 20 elements, $\Delta t = 0.01ms$



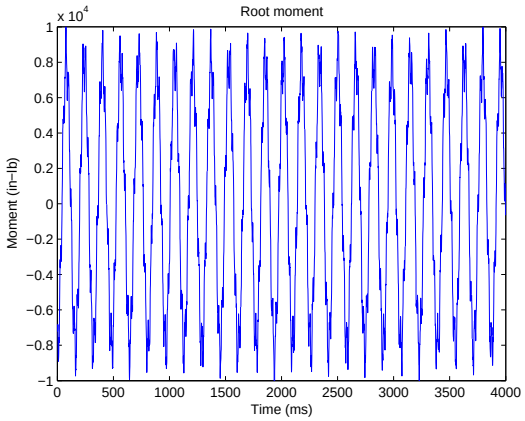
(a) Displacement



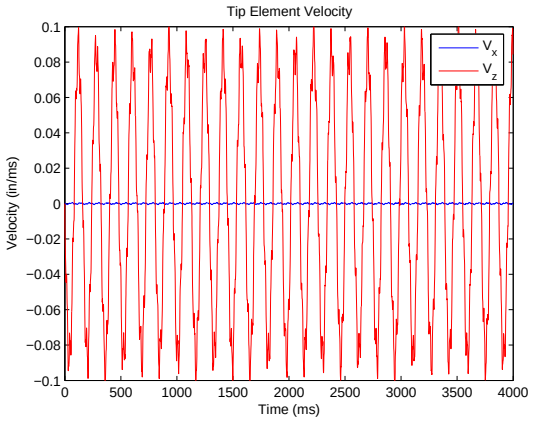
(b) Slope



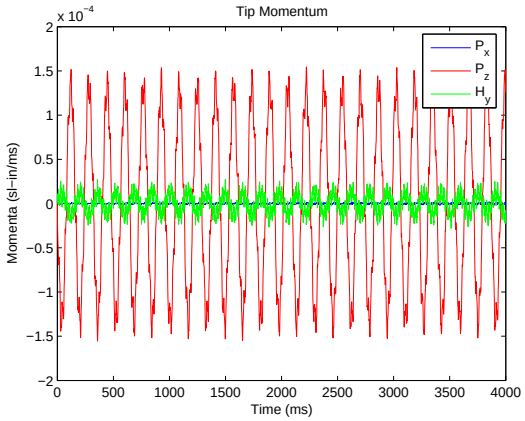
(c) Internal Forces



(d) Internal Moment

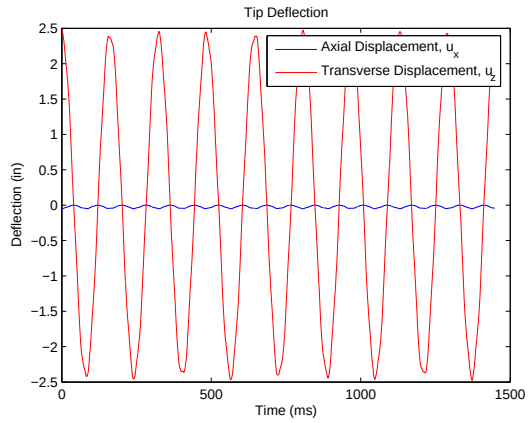


(e) Tip Velocity

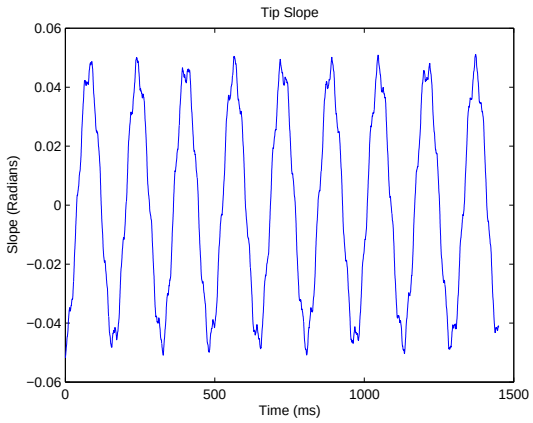


(f) Tip Momenta

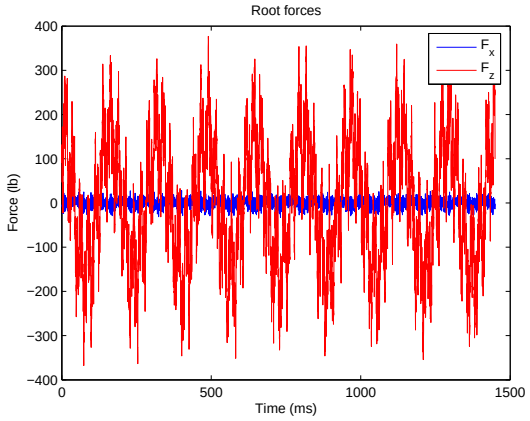
Figure B.10: Bending vibration of a cantilevered beam, 10 elements, $\Delta t = 1ms$



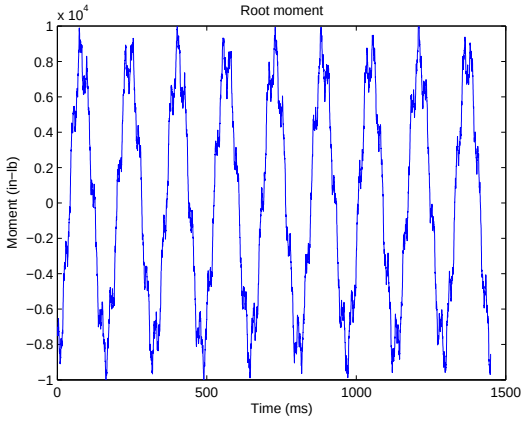
(a) Displacement



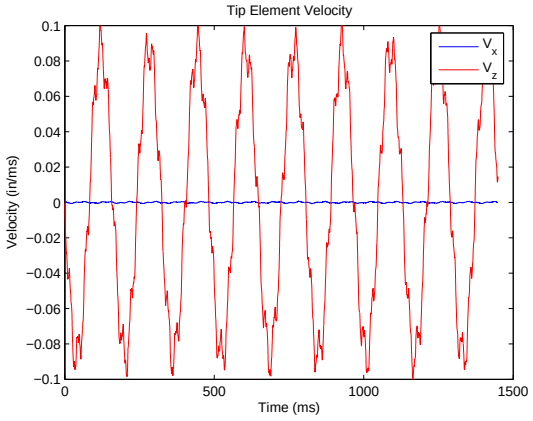
(b) Slope



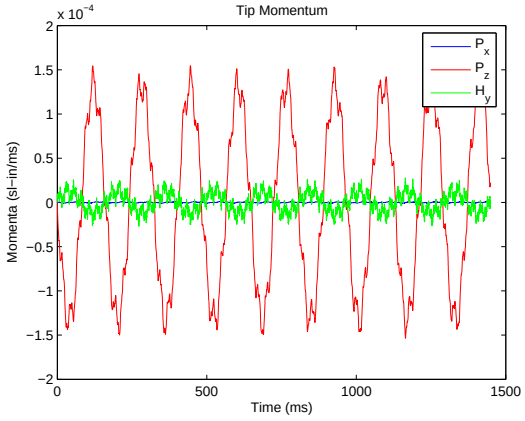
(c) Internal Forces



(d) Internal Moment

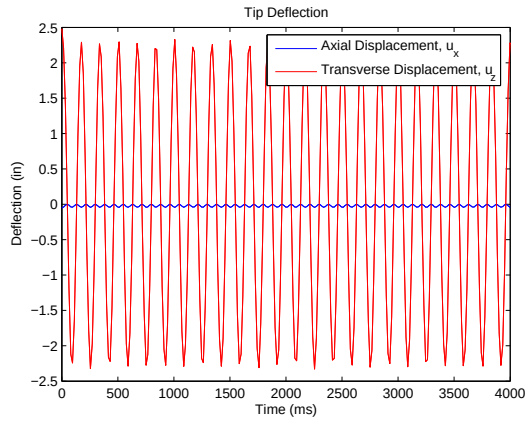


(e) Tip Velocity

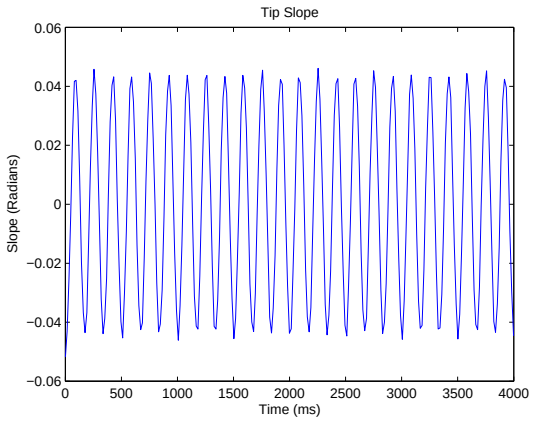


(f) Tip Momenta

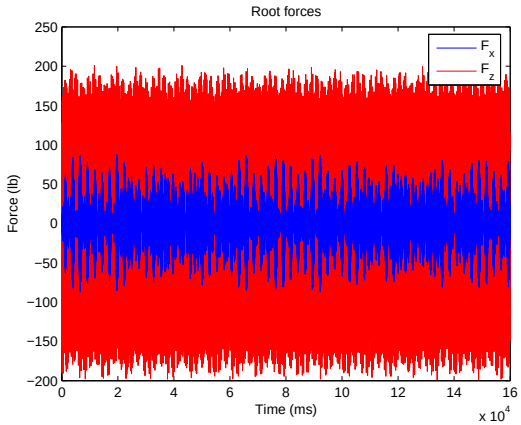
Figure B.11: Bending vibration of a cantilevered beam, 10 elements, $\Delta t = 0.145ms$



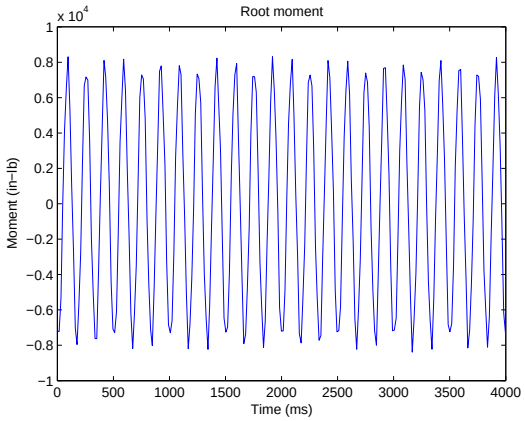
(a) Displacement



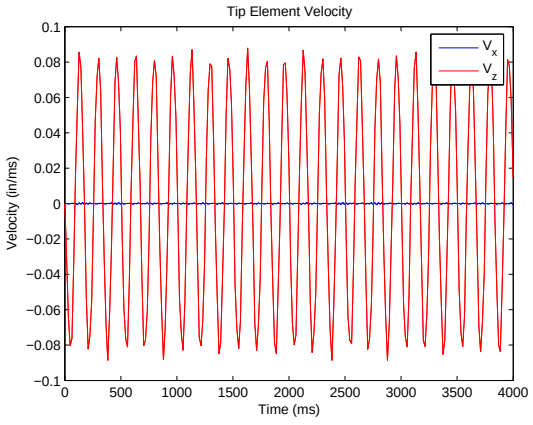
(b) Slope



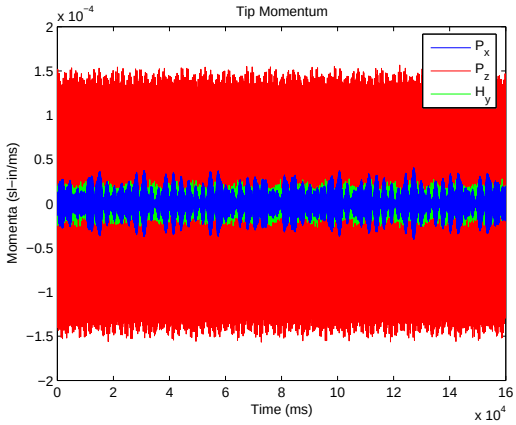
(c) Internal Forces



(d) Internal Moment

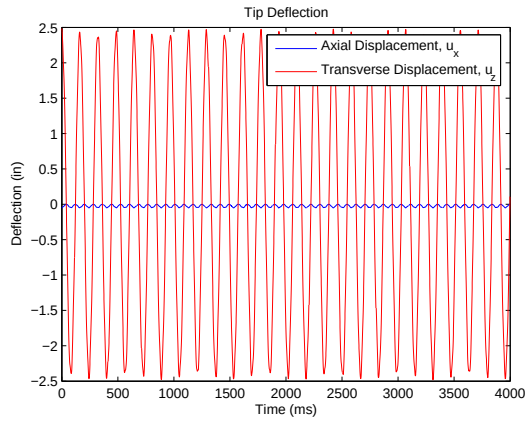


(e) Tip Velocity

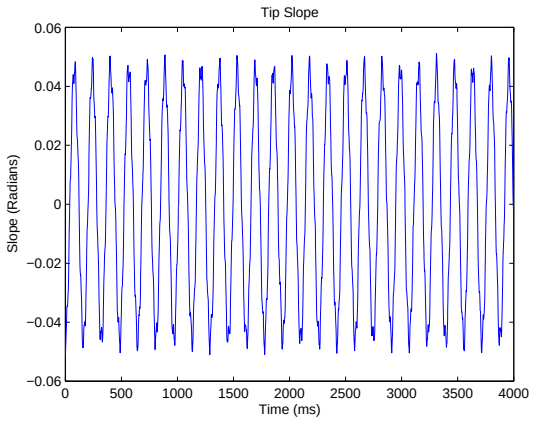


(f) Tip Momenta

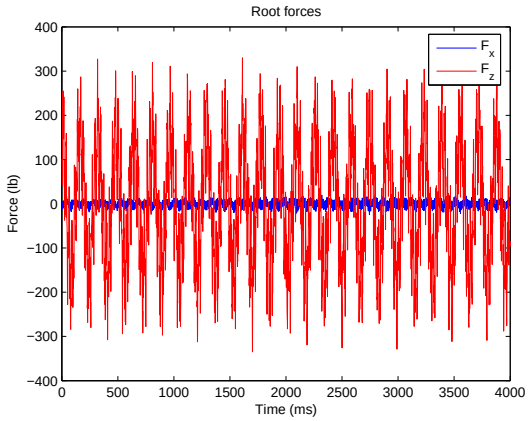
Figure B.12: Bending vibration of a cantilevered beam, 10 elements, $\Delta t = 16ms$



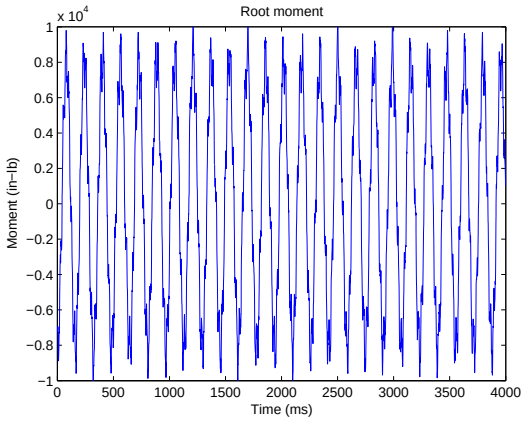
(a) Displacement



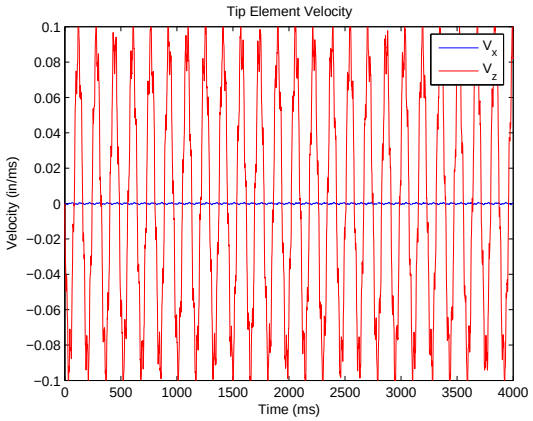
(b) Slope



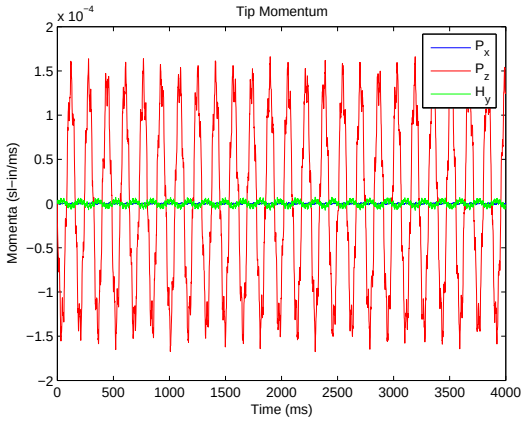
(c) Internal Forces



(d) Internal Moment

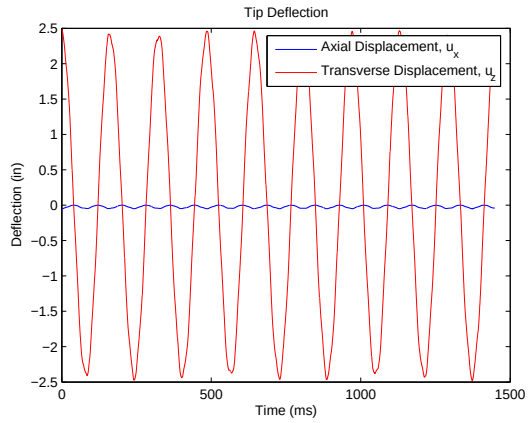


(e) Tip Velocity

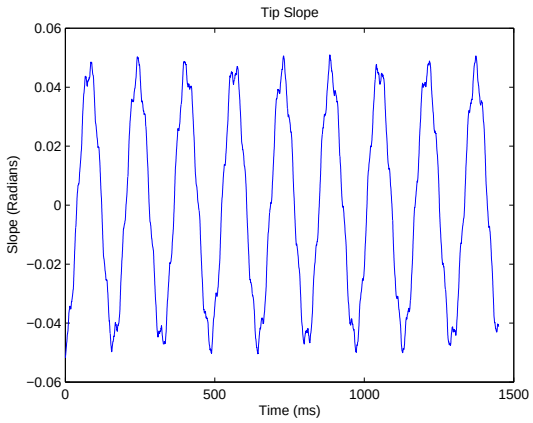


(f) Tip Momenta

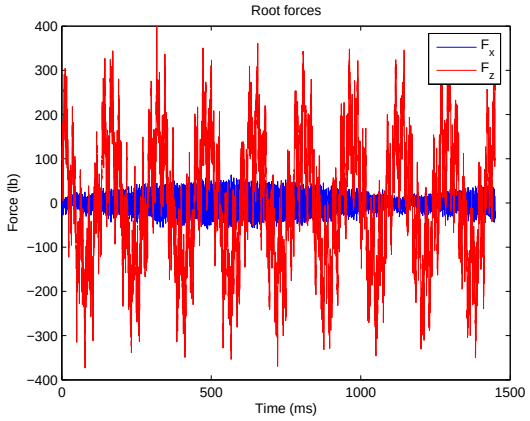
Figure B.13: Bending vibration of a cantilevered beam, 20 elements, $\Delta t = 1ms$



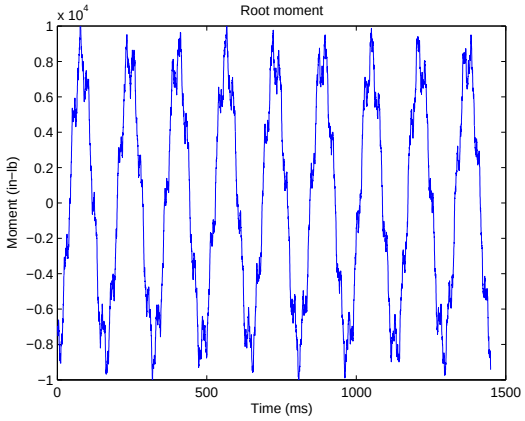
(a) Displacement



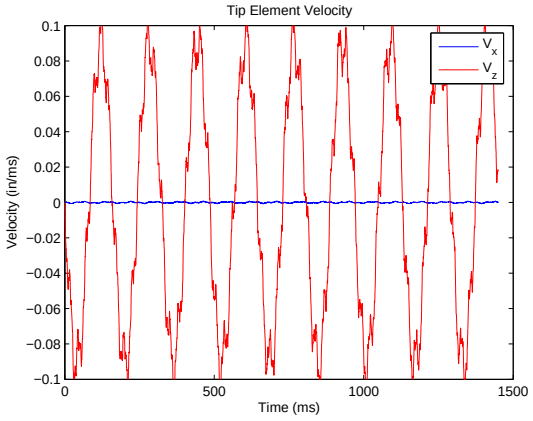
(b) Slope



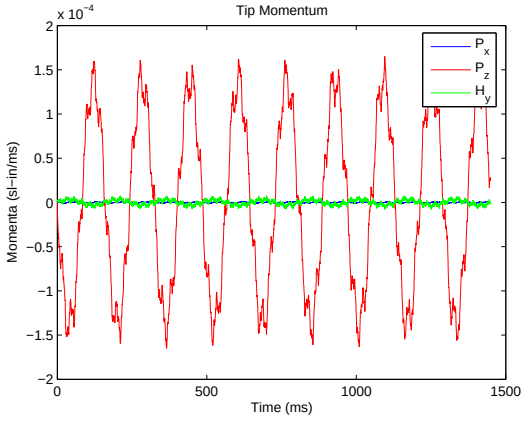
(c) Internal Forces



(d) Internal Moment

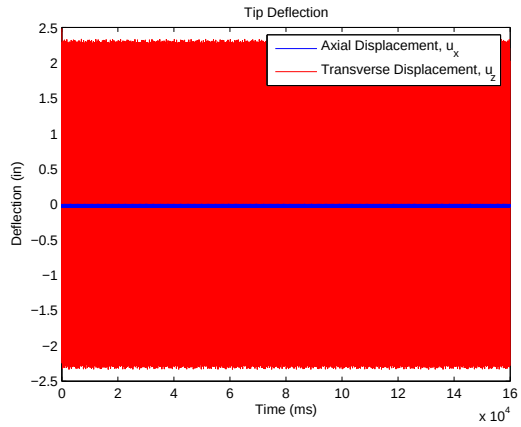


(e) Tip Velocity

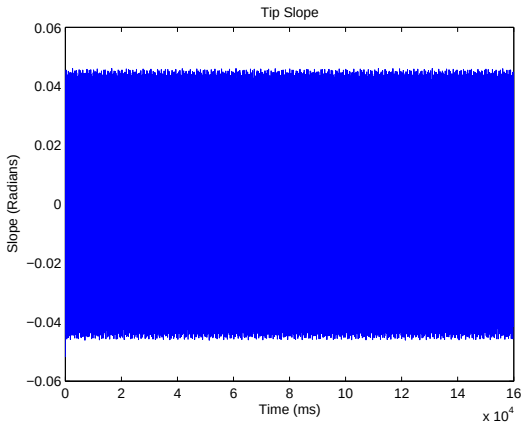


(f) Tip Momenta

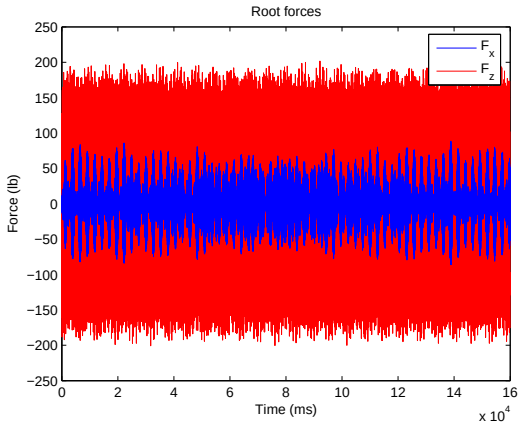
Figure B.14: Bending vibration of a cantilevered beam, 20 elements, $\Delta t = 0.145ms$



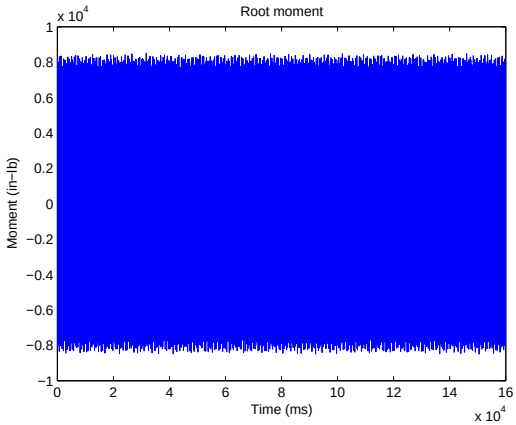
(a) Displacement



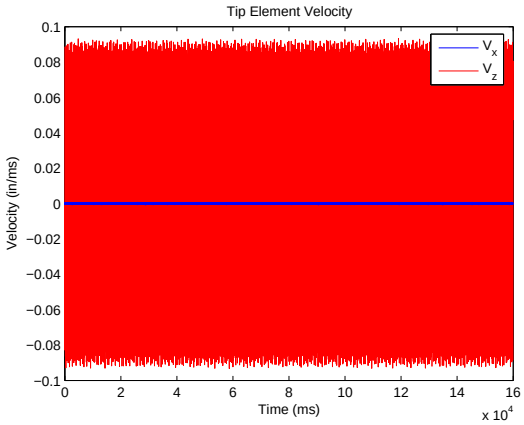
(b) Slope



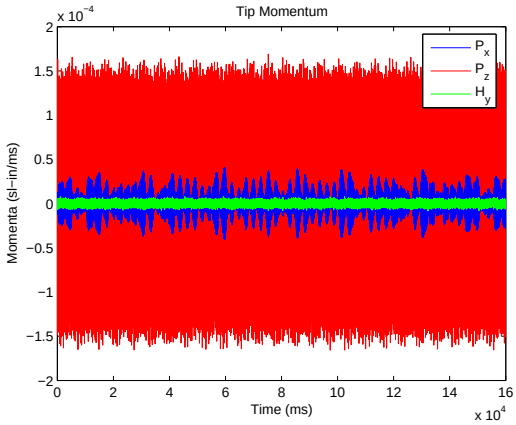
(c) Internal Forces



(d) Internal Moment

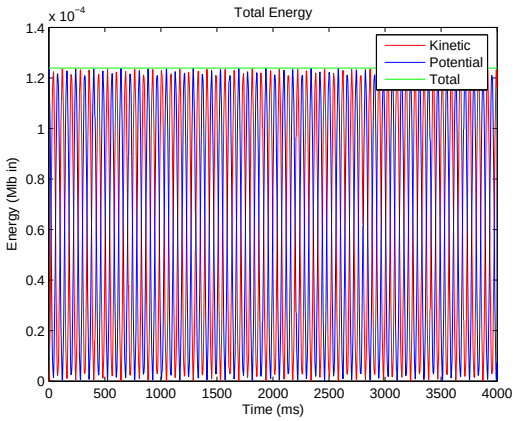


(e) Tip Velocity

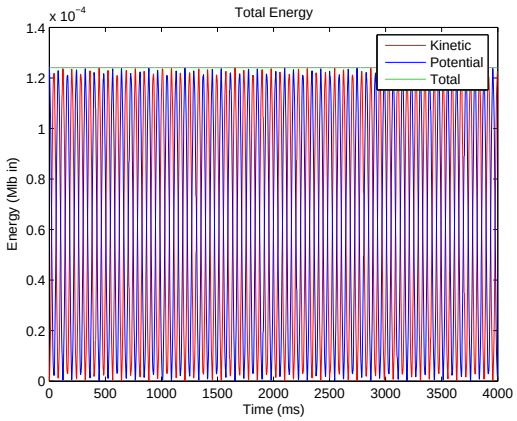


(f) Tip Momenta

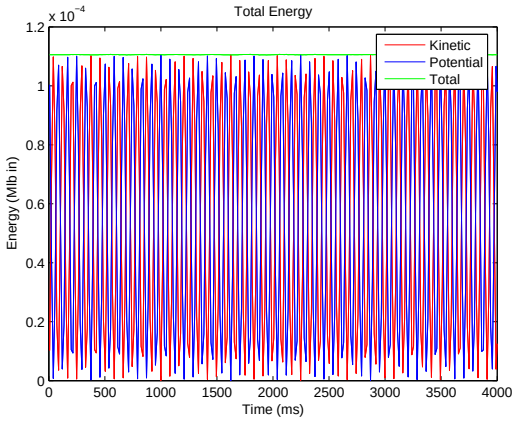
Figure B.15: Bending vibration of a cantilevered beam, 20 elements, $\Delta t = 16ms$



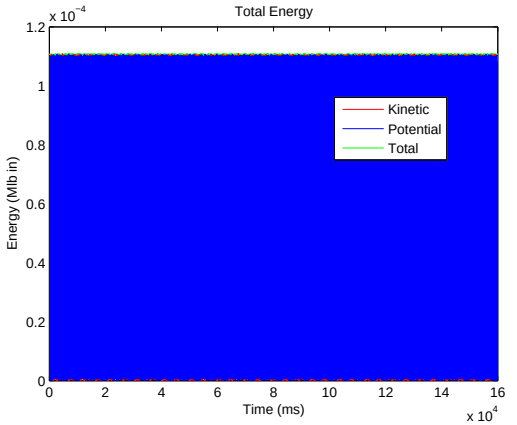
(a) $N_x = 10, \Delta t = 1ms$



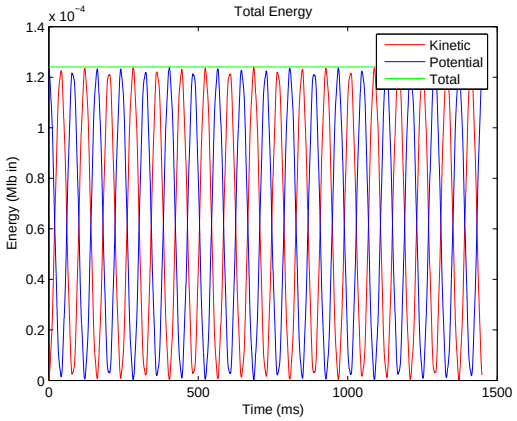
(b) $N_x = 20, \Delta t = 1ms$



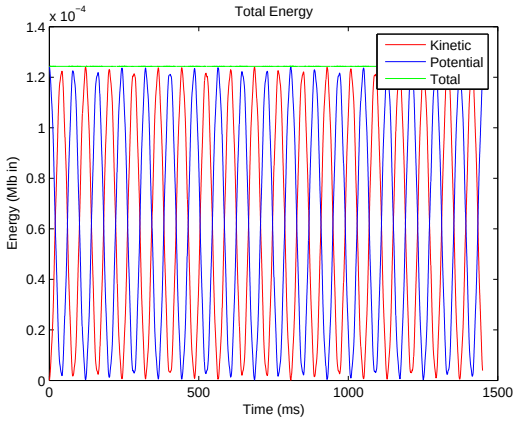
(c) $N_x = 10, \Delta t = 16ms$



(d) $N_x = 20, \Delta t = 16ms$

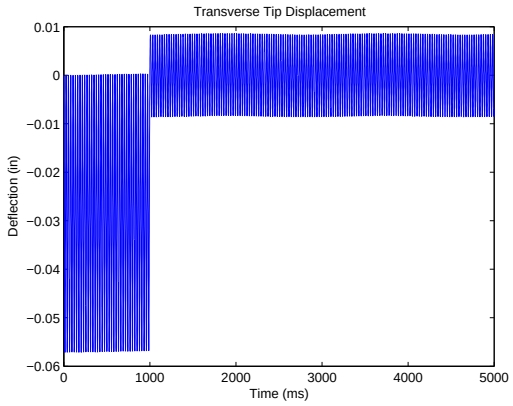


(e) $N_x = 10, \Delta t = 0.145ms$

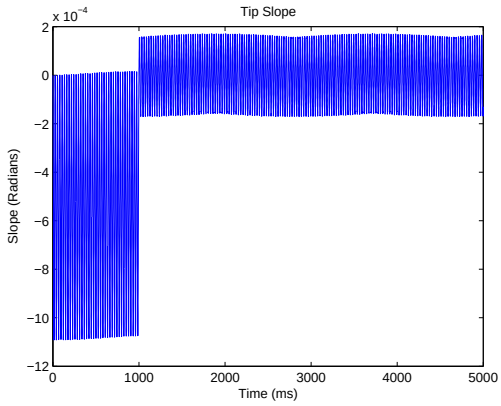


(f) $N_x = 20, \Delta t = 0.145ms$

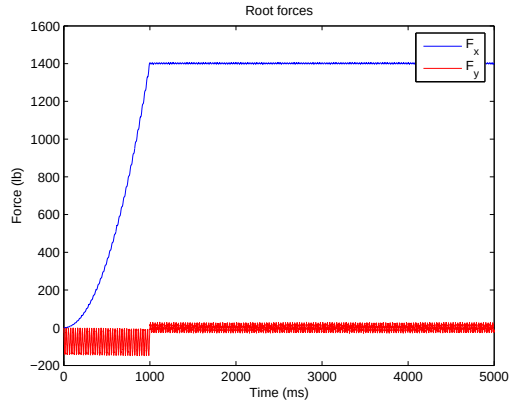
Figure B.16: Conservation of Energy for Bending Vibration



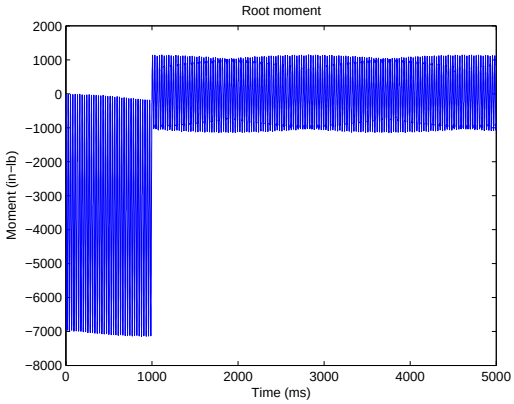
(a) Displacement



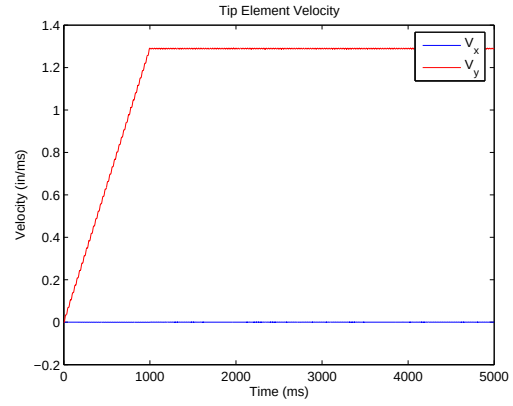
(b) Slope



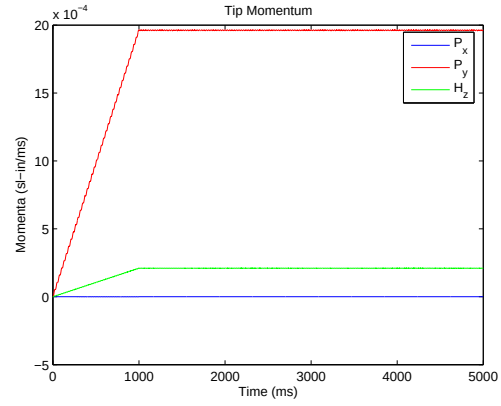
(c) Internal Forces



(d) Internal Moment

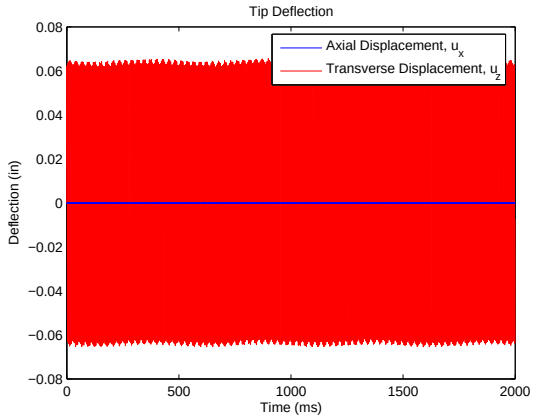


(e) Velocity

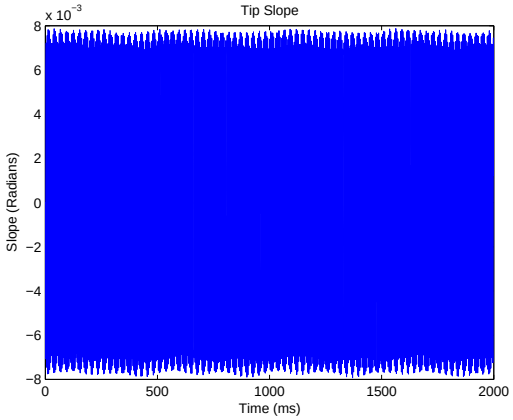


(f) Momenta

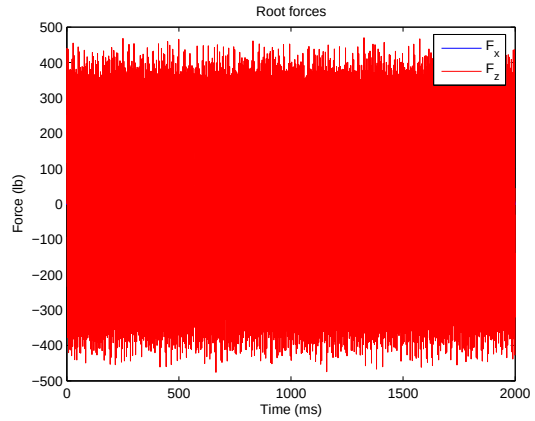
Figure C.1: Spin Up maneuver for a cantilevered beam about the z axis, 10 elements, $\Delta t = 0.5ms$



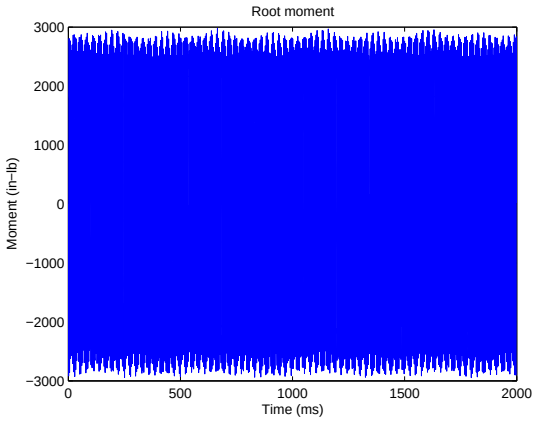
(a) Displacement



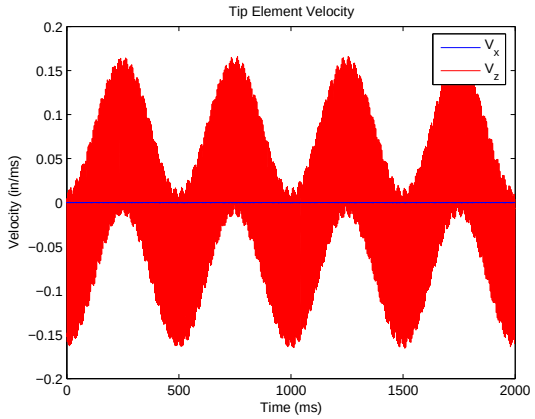
(b) Slope



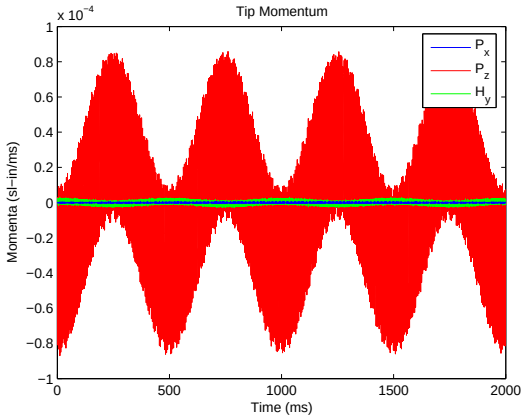
(c) Internal Forces



(d) Internal Moment

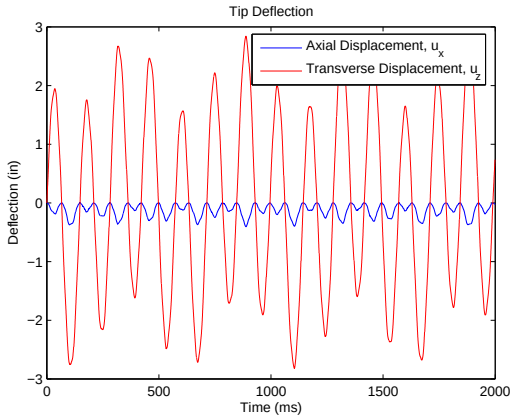


(e) Velocity

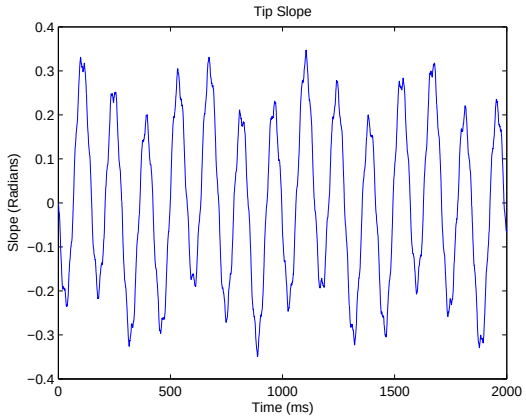


(f) Momenta

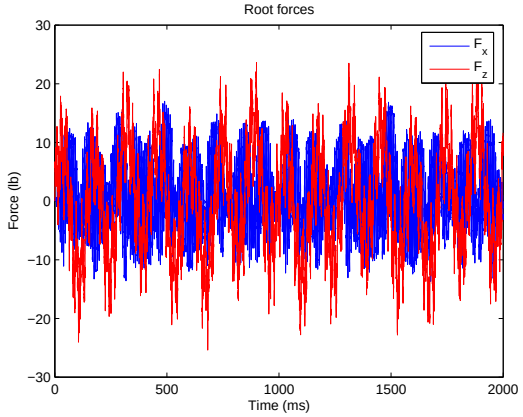
Figure C.2: Flapping maneuver, 10 elements, $\Delta t = 0.2ms$, $\theta_{max} = \pm\pi/6$ at 2Hz, $E = 10e6\,psi$



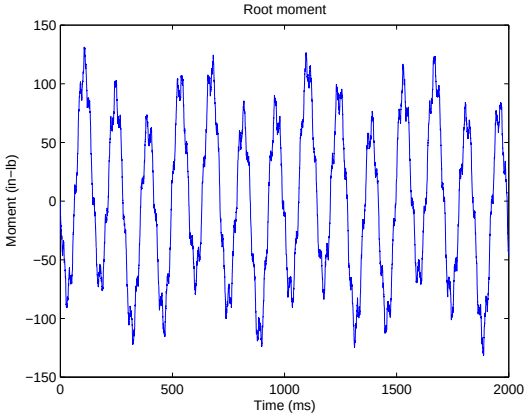
(a) Displacement



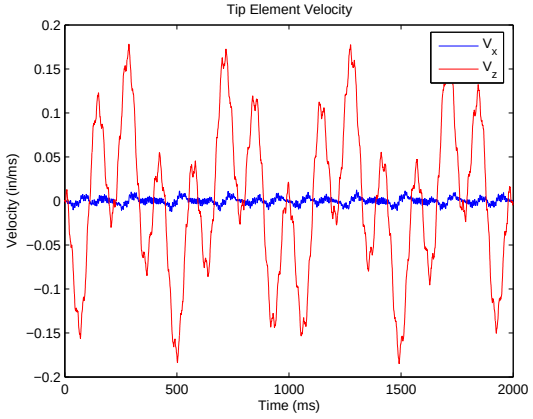
(b) Slope



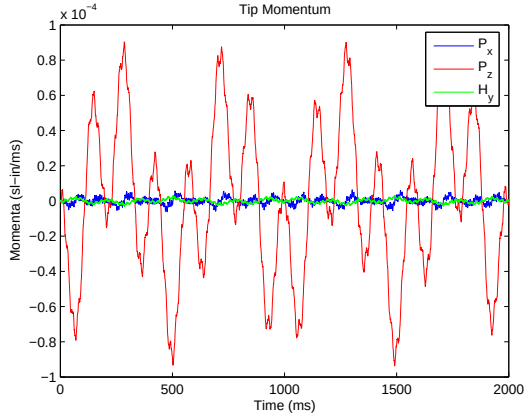
(c) Internal Forces



(d) Internal Moment



(e) Velocity



(f) Momenta

Figure C.3: Flapping maneuver, 10 elements, $\Delta t = 0.2ms$, $\theta_{max} = \pm\pi/6$ at 2Hz, $E = 10e6\,psi$

Appendix D. Matlab Code

This Appendix contains a listing of the MATLAB® files used in this research. The list includes all of the files used in the algorithm and samples of input files and post processors.

D.1 Executable.m

The executable file is used to specify the inputs for the problem and the mesh size. In this case the incremental loading routine is commented out because it is not used. For highly nonlinear problems with non-zero initial conditions, it is uncommented and the steady routine is commented out (only one of the two is required). For static and steady-state problems the dynamic routine is also turned off. The user also specifies the number of elements, step size, damping factor (if used), and output file names.

```
%%-----------------------------------------------------------------------%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SPACE TIME FINITE ELEMENT SOLVER FOR AN ELASTIC BEAM USING  %
%               HAMILTON'S WEAK PRINCIPLE                     %
%                                                             %
% 1st Lt Elliott Leigh                                        %
% Air Force Institute of Technology                           %
% Created 14 Nov 2005                                         %
% Last Modified 19 Jan 2006                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% The following m-files are required to run this program:
%   Geomat.m      ~ contains geometric and material properties
%   Simparams.m   ~ contains boundary conditions, applied loads, velocities
%   FEStatic.m    ~ finite element routine for steady problems
%   FEIncStatic.m ~ incremental finite element routine for steady problems
%   FEDynamic.m   ~ finite element routine for time accurate problems
%   Static.m      ~ equation solver called by FEStatic
%   IStatic.m     ~ equation solver called by FEIncStatic
%   Dynamic.m     ~ equation solver called by FEDynamic
%   ImportStatic.m ~ postprocesses steady data
%   ImportIStatic.m~ postprocesses incremental steady data
%   ImportDynamic.m~ postprocesses time accurate data
%   Milenkovic.m  ~ calculates rotation and direction cosine matrices
%   tilde.m       ~ cross product operator
```

```
clear;clc;clear all; global SIMS

%%%%%%%%%%%%%%%%%%%%%% PROBLEM SET UP %%%%%%%%%%%%%%%%%%%%%%%
SIMS.num_x=1;                    % Number of spatial elements
SIMS.num_t=10000;                % Number of time steps
SIMS.num_q=10;                   % Number incremental load steps
Geomat                           % Geometric/material properties
SIMS.dx=GEOM.length/SIMS.num_x; % Step size in space (inches)
SIMS.dt=0.1                      % Step size in time (milliseconds)
SIMS.time=SIMS.num_t*SIMS.dt;    % Run time of simulation (milliseconds)
Simparams                        % Simulation parameters
SIMS.edf=0;                      % Damping factor (force)
SIMS.edm=0;                      % Damping factor (moment)
SIMS.sample=1;                   % Sampling rate for output data
SName=sprintf('XStatic.dat');    % Output name for static state vector
DName=sprintf('dynamic.dat');    % Output name for dynamic state vector

%%%%%%%%%%%%%%%%%%%%%%%% ANALYSIS %%%%%%%%%%%%%%%%%%%%%%%%%%
ts=cputime;                      % Starts clock to track runtime
FEStatic                         % Steady analysis
% FEIncStatic                    % Incremented steady analysis
% FEDynamic                      % Dynamic analysis
tf=cputime;                      % Stops clock
runtime=tf-ts                    % Records runtime

%%%%%%%%%%%%%%%%%%%%%% POST PROCESS %%%%%%%%%%%%%%%%%%%%%%%%%
% ImportStatic                     % requires XStatic.dat,geometry,Simparams
% ImportIStatic                    % requires XIStatic.dat,geometry,Simparams
% ImportDynamic                    % requires XDynamic.dat +one of the above
%%-------------------------------------------------------------------------%%
```

D.2 Geomat.m

The user specifies geometric and material properties by generating a separate file which is called in the main program. There are separate files for each of the geometries presented in the research. The following sample is the input for the 72-inch aluminum beam used in the follower force problem and spin up maneuver. This approach is the least efficient for isotropic beams because it involves excessive looping and MATLAB® is well suited for vectorized operations. However, it is intended to represent all of the possible parameters that may be required in more complex problems.

```
%%-------------------------------------------------------------------------%%
```

```
%Geometric and Material Properties
global GEOM;
global MATL;
% Geometric Properties
GEOM.length=72;                                    %length of beam (inches)
for n=1:SIMS.num_x;
    GEOM.k_b(:,n)=[0;0;0];                         %undeformed curvature
    GEOM.height(n)=1;                              %height of cross section
    GEOM.width(n)=6;                               %width of cross section
    GEOM.area(n)=GEOM.height(n)*GEOM.width(n);     %cross section area
    GEOM.r_cg_b(:,n)=[0;0;0];                      %center of mass offset
    GEOM.e1=[1;0;0];                               %beam's axial vector
    GEOM.J(n)=3/2;                                 %torsional rigidity
    GEOM.K_1(n)=GEOM.area(n);                      %shear area y
    GEOM.K_2(n)=GEOM.area(n);                      %shear area z
    GEOM.i_b(:,:,n)=[6 0 0;                        %moment of area matrix
                     0 1/2 0;
                     0 0 18];
    % Material Properties
    MATL.mass(n)=.098/386.4*GEOM.area(n);          %mass per unit length (sln)
    MATL.E(n)=10;                                  %elastic modulus (Msi)
    MATL.Poisson(n)=.35;                           %Poisson's ratio
    MATL.G(n)=MATL.E(n)/2/(1+MATL.Poisson(n));     %shear modulus (Msi)
    % Constitutive properties for an isotropic beam
    MATL.A(:,:,n)=[MATL.E(n)*GEOM.area(n) 0 0;
                   0 MATL.G(n)*GEOM.K_1(n) 0;
                   0 0 MATL.G(n)*GEOM.K_2(n)];
    MATL.B(:,:,n)=zeros(3,3);
    MATL.D(:,:,n)=[MATL.G(n)*GEOM.J(n) 0 0;
                   0 MATL.E(n)*GEOM.i_b(2,2,n) 0;
                   0 0 MATL.E(n)*GEOM.i_b(3,3,n)];
    % Moment of inertia per unit length about root
    % Moment of inertia per unit length about root
    MATL.I_b(:,:,n)=MATL.mass(n)*...
              [1/12*(GEOM.height(n)^2+GEOM.width(n)^2) 0 0;
              0 1/12*(GEOM.height(n)^2+(GEOM.length/SIMS.num_x)^2) 0;
              0 0 1/12*(GEOM.width(n)^2+(GEOM.length/SIMS.num_x)^2)];
end
%%-------------------------------------------------------------------------%%
```

D.3 Simparams.m

The boundary conditions, prescribed motion, and applied loads are unique to each problem and specified in a separate file which is called by the main program.

The following is a sample file for a cantilevered beam subject to a steady rotation of 3 Hz. It was used in problems involving steady rotations.

```
%%-----------------------------------------------------------------------%%
% Simulation Parameters
%% All forces are in Mlbs, all moments in Mlb-in, all velocities in in/ms
global RIGD; global LOAD; global BC;
% Initial Conditions (for static problems and initial state of simulations)
BC.F2_B0(:,SIMS.num_x)=[0;0;0];    % initial bc's ~ tip force, B basis(Mlb)
BC.M2_B0(:,SIMS.num_x)=[0;0;0];    % " "
BC.q4_b0(:,1)=[0;0;0];             % initial bc's ~ cantilevered
BC.r4_b0(:,1)=[0;0;0];             % " "
RIGD.w0=[0;0;3*2*pi/1000];         % angular velocity, undeformed (b wrt I)
LOAD.G0=[0;0;0];                   % gravity (b basis)
for n=1:SIMS.num_x
    r(:,n)=[SIMS.dx*(n-1/2);0;0];  % position of beam segment in b
    RIGD.dr0(:,n)=[0;0;0];         % velocity of segment in b frame (in/ms)
    RIGD.v0(:,n)=RIGD.dr0(:,n)+tilde(RIGD.w0(:))*r(:,n);  % in I frame
    if n==SIMS.num_x
        LOAD.f0(:,n)=[0;0;0];     % force per unit length (B basis)
        LOAD.m0(:,n)=[0;0;0];          % moment per unit length (B basis)
    else
        LOAD.f0(:,n)=[0;0;0];          % force per unit length (B basis)
        LOAD.m0(:,n)=[0;0;0];          % moment per unit length (B basis)
    end
end
% Time Varying Conditions (for dynamic problems)
for t=1:SIMS.num_t;
    BC.q4_b(:,1,t)=[0;0;0];                % Cantilevered
    BC.r4_b(:,1,t)=[0;0;0];                % " "

    % Rigid Body Motion and Boundary Conditions
    BC.F2_B(:,SIMS.num_x,t)=[0;0;0];% Tip loads, B basis (follower)
    BC.M2_B(:,SIMS.num_x,t)=[0;0;0];   % " "
    RIGD.w_b(:,t)=[0;0;0];             % angular velocity (b wrt I)
    LOAD.G(:,t)=[0;0;0];               % gravity (b basis)

    for n=1:SIMS.num_x;
        rdot(:,n,t)=[0;0;0];               % velocity of beam segment in b
        RIGD.v_b(:,n,t)=rdot(:,n,t)+tilde(RIGD.w_b(:,t))*r(:,n);  %in I
        % External applied loads
        LOAD.f_B(:,n,t)=[0;0;0];
        LOAD.m_B(:,n,t)=[0;0;0];
    end
end
%%-----------------------------------------------------------------------%%
```

D.4 FEStatic.m

The following script file is the finite element routine for steady-state problems. First, the initial guess is specified (usually zero for all state variables). This guess is supplied to the MATLAB® function fsolve.m, which is used to solve the nonlinear system of equations. Finally, the solution is stored for post-processing and set as the initial conditions for the simulation. Note that in this procedure, the components of the state vector are dependent on the boundary conditions. In this case, the root displacements and tip loads do not appear in the state vector because they are specified in the problem. For a different scenario, they may be unknowns and would appear in the state vector. Regardless, the size of the state vector will remain the same. However, this makes it more difficult to switch between boundary conditions. It is possible but requires more complexity in the code to sort the state vector properly for each type of boundary condition.

```
%%-----------------------------------------------------------------------%%
% Solves the steady boundary value problem
global IC

% Assemble state vector
for n=1:SIMS.num_x;

    % Initial guess for displacements (default zero)
    q0=[0;0;0];
    rho0=[0;0;0];
    [HH0,C0]=Milenkovic(rho0);
    R_CG_B=C0*(GEOM.r_cg_b(:,n));

    % Initial guess for V,W,G,K from q,r (kinematic)
    V0=C0*(RIGD.v0(:,n)+tilde(RIGD.w0)*q0);
    W0=C0*(RIGD.w0);
    G0=C0*(GEOM.e1+tilde(GEOM.k_b(:,n))*q0)-GEOM.e1;
    K0=C0*GEOM.k_b(:,n);

    % Initial guess for P,H,F,M from V,W,G,K (constitutive)
    P0=MATL.mass(n)*V0-MATL.mass(n)*tilde(R_CG_B)*W0;
    H0=MATL.mass(n)*tilde(R_CG_B)*V0+MATL.I_b(:,:,n)*W0;
    F0=MATL.A(:,:,n)*G0+MATL.B(:,:,n)*K0;
    M0=MATL.B(:,:,n)'*G0+MATL.D(:,:,n)*K0;
```

```
    % Assemble (14n variables depend on BC's)
    X0((42*n-41):(42*n-39),1)=q0;            %q
    X0((42*n-38):(42*n-36),1)=rho0;          %r
    X0((42*n-35):(42*n-33),1)=q0;            %q3
    X0((42*n-32):(42*n-30),1)=rho0;          %r3
    X0((42*n-29):(42*n-27),1)=P0;            %P2
    X0((42*n-26):(42*n-24),1)=H0;            %H2
    X0((42*n-23):(42*n-21),1)=q0;            %q2
    X0((42*n-20):(42*n-18),1)=rho0;          %r2
    X0((42*n-17):(42*n-15),1)=F0;            %F1
    X0((42*n-14):(42*n-12),1)=M0;            %M1
    X0((42*n-11):(42*n -9),1)=V0;            %V
    X0((42*n -8):(42*n -6),1)=W0;            %W
    X0((42*n -5):(42*n -3),1)=G0;            %G
    X0((42*n -2):(42*n),1)   =K0;            %K

end

% Solve 14n nonlinear algebraic equations
Options=optimset('Display','off','MaxIter',40000,'TolFun',...
    1e-10,'TolX',1e-10,'MaxFunEvals',...
    40000,'NonlEqnAlgorithm','gn','Jacobian','on');
[XStatic,FVAL0,EXITFLAG0,OUTPUT0]=fsolve(@Static,X0,Options);

% Assign steady solution as initial guess for simulation
X0=XStatic;
% Print to file
dlmwrite(SName,XStatic,'\t');
% Assign initial conditions for time accurate simulation
for n=1:SIMS.num_x;
    IC.q1_b(:,n,1)=XStatic((42*n-35):(42*n-33),1);
    IC.r1_b(:,n,1)=XStatic((42*n-32):(42*n-30),1);
    IC.P1_B(:,n,1)=XStatic((42*n-29):(42*n-27),1);
    IC.H1_B(:,n,1)=XStatic((42*n-26):(42*n-24),1);
end
%%-------------------------------------------------------------------------%%
```

D.5 Static.m

The steady-state solver uses fsolve.m, which requires a user specified function for calculating the residuals of the nonlinear equations. This function contains equations (3.92) through (3.105) and the constraint from (3.106). Variables in the $14N_x$ equations are assigned according to their location in the state vector. The bulk of the

code is associated with calculating the Jacobian, which is a matrix of size $42N_X$ by $42N_X$, representing the partial derivative of each equation with respect to each variable. The calculus used in developing the Jacobian takes too much space to present in this document, however the final answers are presented in the code. While an analytical Jacobian does not need to be supplied, it allows the program to run much more efficiently than having MATLAB® calculate it with finite difference methods.

```
%%---------------------------------------------------------------------%%
function [residual,Jacobian]=Static(X0)
% Minimizes residuals of 14n equations to determine initial conditions

global GEOM  % geometric properties
global MATL  % material properties
global RIGD  % rigid body motion
global LOAD  % distributed applied loads
global SIMS  % mesh parameters
global BC    % boundary conditions

%Initialize Analytical Jacobian
Jacobian=zeros(42*SIMS.num_x,42*SIMS.num_x);

% Calculate residuals and Jacobian using X0 as the guess for the unknowns
for n=1:SIMS.num_x;

    % Boundary conditions and shared edges (problem dependent)
    if n == 1,
        q4=BC.q4_b0(:,1);                       %Boundary condition
        rho4=BC.r4_b0(:,1);                     % " "
    else
        q4=X0((42*(n-1)-23):(42*(n-1)-21),1);   %Shared edge
        rho4=X0((42*(n-1)-20):(42*(n-1)-18),1); %q4=q2 for prev elem
    end
    if n == SIMS.num_x,
        F2=BC.F2_B0(:,SIMS.num_x);              %Boundary condition
        M2=BC.M2_B0(:,SIMS.num_x);              % " "
    else
        F2=X0((42*(n+1)-17):(42*(n+1)-15),1);   %Shared edge
        M2=X0((42*(n+1)-14):(42*(n+1)-12),1);   %F2=F1 for next elem
    end

    % Substitute guess for unknowns (14n properties dependent on problem)
    q   =X0((42*n-41):(42*n-39),1);
    rho =X0((42*n-38):(42*n-36),1);
```

```
q3  =X0((42*n-35):(42*n-33),1);
rho3=X0((42*n-32):(42*n-30),1);
P2  =X0((42*n-29):(42*n-27),1);
H2  =X0((42*n-26):(42*n-24),1);
q2  =X0((42*n-23):(42*n-21),1);
rho2=X0((42*n-20):(42*n-18),1);
F1  =X0((42*n-17):(42*n-15),1);
M1  =X0((42*n-14):(42*n-12),1);
V   =X0((42*n-11):(42*n -9),1);
W   =X0((42*n -8):(42*n -6),1);
G   =X0((42*n -5):(42*n -3),1);
K   =X0((42*n -2):(42*n),1)   ;

% For steady problems, displacements and momenta are constant over time
q1  =q3;  % Enforce kinematics as functions of rigid body motion
rho1=rho3;%
P1  =P2;  % Enforce momenta as constant functions of kinematics
H1  =H2;  %

%Assign intermediate variables
dx=SIMS.dx;
dt=1;     % Auto for steady analysis
mass=MATL.mass(n);
P=1/2*(P1+P2);
H=1/2*(H1+H2);
F=1/2*(F1+F2);
M=1/2*(M1+M2);
[HH,C]=Milenkovic(rho);
g=C*LOAD.G0;              %used in deformed coordinate system
r_cg=C*GEOM.r_cg_b(:,n); %used in deformed coordinate system
f=LOAD.f0(:,n)+mass*g;
m=LOAD.m0(:,n)+mass*tilde(r_cg)*g;
v=RIGD.v0(:,n);
w=RIGD.w0;
k=GEOM.k_b(:,n);
I=MATL.I_b(:,:,n);
A=MATL.A(:,:,n);
B=MATL.B(:,:,n);
D=MATL.D(:,:,n);
e1=GEOM.e1;

%Evaluate 14 equations using guess
delq=dx/4*(P1-P2)-dx*dt/4*tilde(W)*P+dt/4*(F2-F1)+...
    dx*dt/4*tilde(K)*F+dx*dt/4*f;
delp=dx/4*(H1-H2)-dx*dt/4*tilde(W)*H-dx*dt/4*tilde(V)*P+...
    dx*dt/4*tilde(e1+G)*F+dx*dt/4*tilde(K)*M+dt/4*(M2-M1)+dx*dt/4*m;
```

```
delP1=dt/2*(v+tilde(w)*q-C'*V)+(q-q1);
delP2=dt/2*(v+tilde(w)*q-C'*V)+(q3-q);
delH1=dt/2*inv(HH)*(C*w-W)+(rho-rho1);
delH2=dt/2*inv(HH)*(C*w-W)+(rho3-rho);
delF1=dx/2*(C'*(G+e1)-(e1+tilde(k)*q))+(q4-q);
delF2=dx/2*(C'*(G+e1)-(e1+tilde(k)*q))+(q-q2);
delM1=dx/2*inv(HH)*(K-C*k)+(rho4-rho);
delM2=dx/2*inv(HH)*(K-C*k)+(rho-rho2);
delV=P-mass*V+mass*tilde(r_cg)*W;
delW=H-I*W-mass*tilde(r_cg)*V;
delG=A*G+B*(K-k)-F;
delK=B'*G+D*(K-k)-M;

%Assign residuals
residual((42*n-41):(42*n-39),1)=delq;
residual((42*n-38):(42*n-36),1)=delp;
residual((42*n-35):(42*n-33),1)=delP1;
residual((42*n-32):(42*n-30),1)=delP2;
residual((42*n-29):(42*n-27),1)=delH1;
residual((42*n-26):(42*n-24),1)=delH2;
residual((42*n-23):(42*n-21),1)=delF1;
residual((42*n-20):(42*n-18),1)=delF2;
residual((42*n-17):(42*n-15),1)=delM1;
residual((42*n-14):(42*n-12),1)=delM2;
residual((42*n-11):(42*n- 9),1)=delV;
residual((42*n- 8):(42*n- 6),1)=delW;
residual((42*n- 5):(42*n- 3),1)=delG;
residual((42*n- 2):(42*n),1)   =delK;

%Define individual vector elements used in calculating the Jacobian
r1=X0((42*n-38),1); %rotation parameter 1- from rho bar
r2=X0((42*n-37),1); %rotation parameter 2- from rho bar
r3=X0((42*n-36),1); %rotation parameter 3- from rho bar
V1=X0((42*n-11),1); %velocity 1
V2=X0((42*n-10),1); %velocity 2
V3=X0((42*n- 9),1); %velocity 3
G1=X0((42*n-5),1)+1;%strain 1 + e1
G2=X0((42*n-4),1);  %strain 2
G3=X0((42*n-3),1);  %strain 3
K1=X0((42*n-2),1);  %curvature 1
K2=X0((42*n-1),1);  %curvature 2
K3=X0((42*n),1);    %curvature 3
w1=w(1);            %prescribed angular velocity 1
w2=w(2);            %prescribed angular velocity 2
w3=w(3);            %prescribed angular velocity 3
k1=k(1);            %initial twist
```

```
k2=k(2);            %initial curvature 2
k3=k(3);            %initial curvature 3
W1=X0(42*n -8,1);   %angular velocity 1
W2=X0(42*n -7,1);   %angular velocity 2
W3=X0(42*n -6,1);   %angular velocity 3

%Define Jacobian Components from rotation parameters:

%Partial [C'V] / Partial r
JAden=(16+r1^2+r2^2+r3^2)^3;
JA11=32*(r1^3*(-r3*V2+r2*V3)-12*r1^2*(r2*V2+r3*V3)+4*(16+r2^2+r3^2)*...
    (r2*V2+r3*V3)+r1*(r2^2*(16*V1-r3*V2)+r3*(16*r3*V1+48*V2-r3^2*...
    V2)+r2^3*V3+r2*(-48+r3^2)*V3));
JA21=-16*(-8*r2^3*V1-2*r1^3*(r3*V1+8*V2)-2*r1*(-48*r3*V1+r3^3*V1-...
    128*V2+8*r3^2*V2)-96*r1^2*V3+r1^4*V3-r2^4*V3-(-256+r3^4)*V3+...
    8*r2*((-16+3*r1^2-r3^2)*V1+4*r1*r3*V3)-2*r2^2*(r1*r3*V1-8*r1*...
    V2+r3^2*V3));
JA31=16*(r1^4*V2-24*r1^2*(r3*V1+4*V2)-(16+r2^2+r3^2)*(-8*r3*V1+(-16+...
    r2^2)*V2+r3^2*V2)-2*r1^3*(r2*V1-8*V3)-2*r1*(r2^3*V1+r2*((-48+...
    r3^2)*V1+16*r3*V2)-8*r2^2*V3+8*(16+r3^2)*V3));
JA12=-16*(2*r2^3*(-8*V1+r3*V2)-r2^4*V3+24*r2^2*(r1*V2+4*V3)+(16+...
    r1^2+r3^2)*(-8*r1*V2+r1^2*V3+(-16+r3^2)*V3)+2*r2*(8*(16+r1^2-...
    r3^2)*V1+r3*((-48+r1^2+r3^2)*V2+16*r1*V3)));
JA22=-32*(r1^3*(-4*V1+r2*V3)-r1^2*(r2*r3*V1+16*r2*V2+4*r3*V3)-r3*...
    (r2^3*V1+r2*((-48+r3^2)*V1+16*r3*V2)-12*r2^2*V3+4*(16+r3^2)*...
    V3)+r1*(4*(-16+3*r2^2-r3^2)*V1+r2*(-48+r2^2+r3^2)*V3));
JA32=16*((-256+r1^4+96*r2^2-r2^4-32*r1*r2*r3+2*r1^2*r3^2+r3^4)*...
    V1+2*(r1^3*r2*V2+r1*r2*(-48+r2^2+r3^2)*V2+4*r1^2*(r3*V2+2*r2*...
    V3)+4*((16-3*r2^2)*r3*V2+r3^2*V2+2*r2*(-16+r2^2)*V3-2*r2*r3^2*...
    V3)));
JA13=16*(-r3^4*V2+24*r3^2*(4*V2-r1*V3)+(16+r1^2+r2^2)*((-16+...
    r1^2+r2^2)*V2+8*r1*V3)+2*r3^3*(8*V1+r2*V3)-2*r3*(8*(16+...
    r1^2-r2^2)*V1-r2*(-16*r1*V2+r1^2*V3+(-48+r2^2)*V3)));
JA23=-16*((-256+r1^4+2*r1^2*r2^2+r2^4+32*r1*r2*r3+96*r3^2-r3^4)*...
    V1+2*(-4*r2*(16+r1^2+r2^2)*V3+12*r2*r3^2*V3+r3^3*(-8*V2+r1*...
    V3)+r3*(-8*(-16+r1^2-r2^2)*V2+r1*(-48+r1^2+r2^2)*V3)));
JA33=32*(r1^3*(4*V1+r3*V2)+r1*(4*(16+r2^2-3*r3^2)*V1+r3*(-48+...
    r2^2+r3^2)*V2)+r1^2*(-r2*r3*V1+4*r2*V2+16*r3*V3)+r2*(-r3^3*...
    V1+4*(16+r2^2)*V2-12*r3^2*V2+r3*(48*V1-r2^2*V1+16*r2*V3)));
JA=1/JAden*[JA11 JA12 JA13;
            JA21 JA22 JA23;
            JA31 JA32 JA33];

%Partial [(H')^-1 w] / Partial r
JB11=r1*w1+r2*w2+r3*w3;
JB21=r2*w1-r1*w2+4*w3;
```

```
JB31=r3*w1-4*w2-r1*w3;
JB12=-r2*w1+r1*w2-4*w3;
JB22=r1*w1+r2*w2+r3*w3;
JB32=4*w1+r3*w2-r2*w3;
JB13=-r3*w1+4*w2+r1*w3;
JB23=-4*w1-r3*w2+r2*w3;
JB33=r1*w1+r2*w2+r3*w3;
JB=1/8*[JB11 JB12 JB13;
        JB21 JB22 JB23;
        JB31 JB32 JB33];

%Partial [(H)^-1 W] / Partial r
JC11=r1*W1+r2*W2+r3*W3;
JC21=-4*W3-r1*W2+r2*W1;
JC31=4*W2-r1*W3+r3*W1;
JC12=4*W3-r1*W2-r2*W1;
JC22=r1*W1+r2*W2+r3*W3;
JC32=-4*W1-r2*W3+r3*W2;
JC13=-4*W2+r1*W3-r3*W1;
JC23=4*W1+r2*W3-r3*W2;
JC33=r1*W1+r2*W2+r3*W3;
JC=1/8*[JC11 JC12 JC13;
        JC21 JC22 JC23;
        JC31 JC32 JC33];

%Partial [C'(G+e1)] / Partial r
JDden=(16+r1^2+r2^2+r3^2)^3;
JD11=32*(r1^3*(-r3*G2+r2*G3)-12*r1^2*(r2*G2+r3*G3)+4*(16+r2^2+...
    r3^2)*(r2*G2+r3*G3)+r1*(r2^2*(16*G1-r3*G2)+r3*(16*r3*G1+48*...
    G2-r3^2*G2)+r2^3*G3+r2*(-48+r3^2)*G3));
JD21=-16*(-8*r2^3*G1-2*r1^3*(r3*G1+8*G2)-2*r1*(-48*r3*G1+r3^3*...
    G1-128*G2+8*r3^2*G2)-96*r1^2*G3+r1^4*G3-r2^4*G3-(-256+r3^4)*...
    G3+8*r2*((-16+3*r1^2-r3^2)*G1+4*r1*r3*G3)-2*r2^2*(r1*r3*G1-8*...
    r1*G2+r3^2*G3));
JD31=16*(r1^4*G2-24*r1^2*(r3*G1+4*G2)-(16+r2^2+r3^2)*(-8*r3*G1+...
    (-16+r2^2)*G2+r3^2*G2)-2*r1^3*(r2*G1-8*G3)-2*r1*(r2^3*G1+r2*...
    ((-48+r3^2)*G1+16*r3*G2)-8*r2^2*G3+8*(16+r3^2)*G3));
JD12=-16*(2*r2^3*(-8*G1+r3*G2)-r2^4*G3+24*r2^2*(r1*G2+4*G3)+(16+...
    r1^2+r3^2)*(-8*r1*G2+r1^2*G3+(-16+r3^2)*G3)+2*r2*(8*(16+r1^2-...
    r3^2)*G1+r3*((-48+r1^2+r3^2)*G2+16*r1*G3)));
JD22=-32*(r1^3*(-4*G1+r2*G3)-r1^2*(r2*r3*G1+16*r2*G2+4*r3*G3)-r3*...
    (r2^3*G1+r2*((-48+r3^2)*G1+16*r3*G2)-12*r2^2*G3+4*(16+r3^2)*...
    G3)+r1*(4*(-16+3*r2^2-r3^2)*G1+r2*(-48+r2^2+r3^2)*G3));
JD32=16*((-256+r1^4+96*r2^2-r2^4-32*r1*r2*r3+2*r1^2*r3^2+r3^4)*...
    G1+2*(r1^3*r2*G2+r1*r2*(-48+r2^2+r3^2)*G2+4*r1^2*(r3*G2+2*...
    r2*G3)+4*((16-3*r2^2)*r3*G2+r3^2*G2+2*r2*(-16+r2^2)*G3-2*r2*...
```

```
    r3^2*G3)));
JD13=16*(-r3^4*G2+24*r3^2*(4*G2-r1*G3)+(16+r1^2+r2^2)*((-16+r1^2+...
    r2^2)*G2+8*r1*G3)+2*r3^3*(8*G1+r2*G3)-2*r3*(8*(16+r1^2-r2^2)*...
    G1-r2*(-16*r1*G2+r1^2*G3+(-48+r2^2)*G3)));
JD23=-16*((-256+r1^4+2*r1^2*r2^2+r2^4+32*r1*r2*r3+96*r3^2-r3^4)*...
    G1+2*(-4*r2*(16+r1^2+r2^2)*G3+12*r2*r3^2*G3+r3^3*(-8*G2+r1*...
    G3)+r3*(-8*(-16+r1^2-r2^2)*G2+r1*(-48+r1^2+r2^2)*G3)));
JD33=32*(r1^3*(4*G1+r3*G2)+r1*(4*(16+r2^2-3*r3^2)*G1+r3*(-48+r2^2+...
    r3^2)*G2)+r1^2*(-r2*r3*G1+4*r2*G2+16*r3*G3)+r2*(-r3^3*G1+4*...
    (16+r2^2)*G2-12*r3^2*G2+r3*(48*G1-r2^2*G1+16*r2*G3)));
JD=1/JDden*[JD11 JD12 JD13;
            JD21 JD22 JD23;
            JD31 JD32 JD33];

%Partial [(H')^-1 k] / Partial r
JE11=r1*k1+r2*k2+r3*k3;
JE21=r2*k1-r1*k2+4*k3;
JE31=r3*k1-4*k2-r1*k3;
JE12=-r2*k1+r1*k2-4*k3;
JE22=r1*k1+r2*k2+r3*k3;
JE32=4*k1+r3*k2-r2*k3;
JE13=-r3*k1+4*k2+r1*k3;
JE23=-4*k1-r3*k2+r2*k3;
JE33=r1*k1+r2*k2+r3*k3;
JE=1/8*[JE11 JE12 JE13;
        JE21 JE22 JE23;
        JE31 JE32 JE33];

%Partial [(H)^-1 K] / Partial r
JF11=r1*K1+r2*K2+r3*K3;
JF21=-4*K3-r1*K2+r2*K1;
JF31=4*K2-r1*K3+r3*K1;
JF12=4*K3-r1*K2-r2*K1;
JF22=r1*K1+r2*K2+r3*K3;
JF32=-4*K1-r2*K3+r3*K2;
JF13=-4*K2+r1*K3-r3*K1;
JF23=4*K1+r2*K3-r3*K2;
JF33=r1*K1+r2*K2+r3*K3;
JF=1/8*[JF11 JF12 JF13;
        JF21 JF22 JF23;
        JF31 JF32 JF33];

%Fill the n x n Jacobian block with non zero elements (diagonal block)
Jacobian((42*n-35):(42*n-33),(42*n-41):(42*n-39))=...
    dt/2*tilde(w)+eye(3);%delP1/q
Jacobian((42*n-32):(42*n-30),(42*n-41):(42*n-39))=...
```

```
    dt/2*tilde(w)-eye(3);%delP2/q
Jacobian((42*n-23):(42*n-21),(42*n-41):(42*n-39))=...
    -dx/2*tilde(k)-eye(3);%delF1/q
Jacobian((42*n-20):(42*n-18),(42*n-41):(42*n-39))=...
    -dx/2*tilde(k)+eye(3);%delF2/q
Jacobian((42*n-35):(42*n-33),(42*n-38):(42*n-36))=...
    -dt/2*JA;%delP1/rho
Jacobian((42*n-32):(42*n-30),(42*n-38):(42*n-36))=...
    -dt/2*JA;%delP2/rho
Jacobian((42*n-29):(42*n-27),(42*n-38):(42*n-36))=...
    dt/2*(JB-JC)+eye(3);%delH1/rho
Jacobian((42*n-26):(42*n-24),(42*n-38):(42*n-36))=...
    dt/2*(JB-JC)-eye(3);%delH2/rho
Jacobian((42*n-23):(42*n-21),(42*n-38):(42*n-36))=dx/2*JD;%delF1/rho
Jacobian((42*n-20):(42*n-18),(42*n-38):(42*n-36))=dx/2*JD;%delF2/rho
Jacobian((42*n-17):(42*n-15),(42*n-38):(42*n-36))=...
    dx/2*(JF-JE)-eye(3);%delM1/rho
Jacobian((42*n-14):(42*n-12),(42*n-38):(42*n-36))=...
    dx/2*(JF-JE)+eye(3);%delM2/rho
Jacobian((42*n-35):(42*n-33),(42*n-35):(42*n-33))=...
    -eye(3);%delP1/q3-only for steady problems
Jacobian((42*n-32):(42*n-30),(42*n-35):(42*n-33))=eye(3);%delP2/q3
Jacobian((42*n-29):(42*n-27),(42*n-32):(42*n-30))=...
    -eye(3);%delH1/rho3-only for steady problems
Jacobian((42*n-26):(42*n-24),(42*n-32):(42*n-30))=eye(3);%delH2/rho3
Jacobian((42*n-41):(42*n-39),(42*n-29):(42*n-27))=...
    -dx*dt/4*tilde(W);%delq/P2-steady
Jacobian((42*n-38):(42*n-36),(42*n-29):(42*n-27))=...
    -dx*dt/4*tilde(V);%delpsi/P2
Jacobian((42*n-11):(42*n-9),(42*n-29):(42*n-27))=eye(3);%delV/P2-steady
Jacobian((42*n-38):(42*n-36),(42*n-26):(42*n-24))=...
    -dx*dt/4*tilde(W);%delpsi/H2-steady
Jacobian((42*n-8):(42*n-6),(42*n-26):(42*n-24))=eye(3);%delW/H2-steady
Jacobian((42*n-20):(42*n-18),(42*n-23):(42*n-21))=...
    -eye(3);%delF2/q2
Jacobian((42*n-14):(42*n-12),(42*n-20):(42*n-18))=-eye(3);%delM2/rho2
Jacobian((42*n-41):(42*n-39),(42*n-17):(42*n-15))=...
    -dt/4*eye(3)+dx*dt/8*tilde(K);%delq/F1
Jacobian((42*n-38):(42*n-36),(42*n-17):(42*n-15))=...
    dx*dt/8*tilde(e1+G);%delpsi/F1
Jacobian((42*n-5):(42*n-3),(42*n-17):(42*n-15))=-1/2*eye(3);%delG/F1
Jacobian((42*n-38):(42*n-36),(42*n-14):(42*n-12))=...
    -dt/4*eye(3)+dx*dt/8*tilde(K);%delpsi/M1
Jacobian((42*n-2):(42*n),(42*n-14):(42*n-12))=-1/2*eye(3);%delK/M1
Jacobian((42*n-38):(42*n-36),(42*n-11):(42*n-9))=...
    dx*dt/8*tilde(P1+P2);%delpsi/V
```

```
Jacobian((42*n-35):(42*n-33),(42*n-11):(42*n-9))=-dt/2*C';%delP1/V
Jacobian((42*n-32):(42*n-30),(42*n-11):(42*n-9))=-dt/2*C';%delP2/V
Jacobian((42*n-11):(42*n-9),(42*n-11):(42*n-9))=-mass*eye(3);%delV/V
Jacobian((42*n-8):(42*n-6),(42*n-11):(42*n-9))=...
    -mass*tilde(r_cg);%delW/V
Jacobian((42*n-41):(42*n-39),(42*n-8):(42*n-6))=...
    dx*dt/8*tilde(P1+P2);%delq/W
Jacobian((42*n-38):(42*n-36),(42*n-8):(42*n-6))=...
    dx*dt/8*tilde(H1+H2);%delpsi/W
Jacobian((42*n-29):(42*n-27),(42*n-8):(42*n-6))=-dt/2*inv(HH);%delH1/W
Jacobian((42*n-26):(42*n-24),(42*n-8):(42*n-6))=-dt/2*inv(HH);%delH2/W
Jacobian((42*n-11):(42*n-9),(42*n-8):(42*n-6))=...
    mass*tilde(r_cg);%delV/W
Jacobian((42*n-8):(42*n-6),(42*n-8):(42*n-6))=-I;%delW/W
Jacobian((42*n-38):(42*n-36),(42*n-5):(42*n-3))=...
    -dx*dt/8*tilde(F1+F2);%delpsi/G
Jacobian((42*n-23):(42*n-21),(42*n-5):(42*n-3))=dx/2*C';%delF1/G
Jacobian((42*n-20):(42*n-18),(42*n-5):(42*n-3))=dx/2*C';%delF2/G
Jacobian((42*n-5):(42*n-3),(42*n-5):(42*n-3))=A;%delG/G
Jacobian((42*n-2):(42*n),(42*n-5):(42*n-3))=B';%delK/G
Jacobian((42*n-41):(42*n-39),(42*n-2):(42*n))=...
    -dx*dt/8*tilde(F1+F2);%delq/K
Jacobian((42*n-38):(42*n-36),(42*n-2):(42*n))=...
    -dx*dt/8*tilde(M1+M2);%delpsi/K
Jacobian((42*n-17):(42*n-15),(42*n-2):(42*n))=dx/2*inv(HH);%delM1/K
Jacobian((42*n-14):(42*n-12),(42*n-2):(42*n))=dx/2*inv(HH);%delM2/K
Jacobian((42*n-5):(42*n-3),(42*n-2):(42*n))=B;%delG/K
Jacobian((42*n-2):(42*n),(42*n-2):(42*n))=D;%delK/K

%Contributions to off diagonal blocks
if n > 1
    %The n x (n-1) block
    Jacobian((42*n-23):(42*n-21),(42*(n-1)-23):(42*(n-1)-21))=...
        eye(3);%delF1(n)/q2(n-1),ie q4
    Jacobian((42*n-17):(42*n-15),(42*(n-1)-20):(42*(n-1)-18))=...
        eye(3);%delM1(n)/rho2(n-1),ie rho4
end

if n < SIMS.num_x
    %The n x (n+1) block
    Jacobian((42*n-41):(42*n-39),(42*(n+1)-17):(42*(n+1)-15))=...
        dt/4*eye(3)+dx*dt/8*tilde(K);%delq(n)/F(n+1), ie F2
    Jacobian((42*n-38):(42*n-36),(42*(n+1)-17):(42*(n+1)-15))=...
        dx*dt/8*tilde(e1+G);%delpsi(n)/F(n+1)
    Jacobian((42*n-5):(42*n-3),(42*(n+1)-17):(42*(n+1)-15))=...
        -1/2*eye(3);%delG(n)/F(n+1)
```

```
        Jacobian((42*n-38):(42*n-36),(42*(n+1)-14):(42*(n+1)-12))=...
            dt/4*eye(3)+dx*dt/8*tilde(K);%delpsi(n)/M(n+1), ie M2
        Jacobian((42*n-2):(42*n),(42*(n+1)-14):(42*(n+1)-12))=...
            -1/2*eye(3);%delK(n)/M(n+1)
    end

end
%%-------------------------------------------------------------------------%%
```

D.6 FEIncStatic.m

The following file is used for solving steady, nonlinear problems with an incremental loading technique. This is accomplished by updating the initial guess to the solution after each incremental load is solved. If the increments are small enough, the initial guess remains close to the incremental solution. When required, this file is called by the executable in place of FEStatic.m. It increments all prescribed velocities, loads, and boundary conditions (tip loads) linearly according to the number of increments specified and the final value.

```
%%-------------------------------------------------------------------------%%
% Solve the static boundary value problem incrementally
global IC

% Assemble state vector
for n=1:SIMS.num_x;

    % Initial guess for displacements
    q0=[0;0;0];
    r0=[0;0;0];
    [H0,C0]=Milenkovic(r0);
    R_CG_B=C0*(GEOM.r_cg_b(:,n));

    % Initial guess for V,W,G,K from q,r (kinematics)
    V0=C0*(RIGD.v0(:,n)+tilde(RIGD.w0)*q0);
    W0=C0*(RIGD.w0);
    G0=C0*(GEOM.e1+tilde(GEOM.k_b(:,n))*q0)-GEOM.e1;
    K0=C0*GEOM.k_b(:,n);

    % Initial guess for P,H,F,M from V,W,G,K (constitutive properties)
    P0=MATL.mass(n)*V0-MATL.mass(n)*tilde(R_CG_B)*W0;
    H0=MATL.mass(n)*tilde(R_CG_B)*V0+MATL.I_b(:,:,n)*W0;
    F0=MATL.A(:,:,n)*G0+MATL.B(:,:,n)*K0;
```

```
    M0=MATL.B(:,:,n)'*G0+MATL.D(:,:,n)*K0;

    % Assemble (14n variables depend on BC's)
    X0((42*n-41):(42*n-39),1)=q0;            %q
    X0((42*n-38):(42*n-36),1)=r0;            %r
    X0((42*n-35):(42*n-33),1)=q0;            %q3
    X0((42*n-32):(42*n-30),1)=r0;            %r3
    X0((42*n-29):(42*n-27),1)=P0;            %P2
    X0((42*n-26):(42*n-24),1)=P0;            %H2
    X0((42*n-23):(42*n-21),1)=q0;            %q2~ could also guess BC's
    X0((42*n-20):(42*n-18),1)=r0;            %r2~ " "
    X0((42*n-17):(42*n-15),1)=F0;            %F1~ " "
    X0((42*n-14):(42*n-12),1)=M0;            %M1~ " "
    X0((42*n-11):(42*n -9),1)=V0;            %V
    X0((42*n -8):(42*n -6),1)=W0;            %W
    X0((42*n -5):(42*n -3),1)=G0;            %G
    X0((42*n -2):(42*n),1)   =K0;            %K

end

% Solve 14n nonlinear algebraic equations incrementally
for t=1:SIMS.num_q
    Options=optimset('Display','off','MaxIter',40000,'TolFun',...
        1e-10,'TolX',1e-10,'MaxFunEvals',40000,...
        'NonlEqnAlgorithm','gn','Jacobian','on');
    [XIStatic(:,t),FVAL,EXITFLAG]=fsolve(@IStatic,X0,Options,t);
    % Assign solution as next initial guess
    X0=XIStatic(:,t);
end
% Print final solution to file
dlmwrite(SName,XIStatic(:,:),'\t');
% Assign initial conditions for time accurate simulation
for n=1:SIMS.num_x;
    IC.q1_b(:,n,1)=XIStatic((42*n-35):(42*n-33),SIMS.num_q);
    IC.r1_b(:,n,1)=XIStatic((42*n-32):(42*n-30),SIMS.num_q);
    IC.P1_B(:,n,1)=XIStatic((42*n-29):(42*n-27),SIMS.num_q);
    IC.H1_B(:,n,1)=XIStatic((42*n-26):(42*n-24),SIMS.num_q);
end
%%-------------------------------------------------------------------------%%
```

D.7 IStatic.m

Like the steady solver, the incremental loading technique uses fsolve.m to evaluate the system of nonlinear equations. The following file is used to calculate the Jacobian and the residuals for incremental loading problems.

```
%%---------------------------------------------------------------------%%
function [residual,Jacobian]=IStatic(X0,t)
% Minimizes residuals of 14n equations to determine initial conditions

global GEOM  % geometric properties
global MATL  % material properties
global RIGD  % rigid body motion
global LOAD  % distributed applied loads
global SIMS  % mesh parameters
global BC    % boundary conditions

%Initialize Analytical Jacobian
Jacobian=zeros(42*SIMS.num_x,42*SIMS.num_x);

% Calculate residuals and Jacobian using X0 as the guess for the unknowns
for n=1:SIMS.num_x;

    % Boundary conditions and shared edges (problem dependent)
    if n == 1,
        q4=BC.q4_b0(:,1);                       %Boundary condition
        rho4=BC.r4_b0(:,1);                     % " "
    else
        q4=X0((42*(n-1)-23):(42*(n-1)-21),1);   %Shared edge
        rho4=X0((42*(n-1)-20):(42*(n-1)-18),1); %q4=q2 for prev elem
    end
    if n == SIMS.num_x,
        F2=BC.F2_B0(:,SIMS.num_x)*t/SIMS.num_q; %Boundary condition
        M2=BC.M2_B0(:,SIMS.num_x)*t/SIMS.num_q; % " "
    else
        F2=X0((42*(n+1)-17):(42*(n+1)-15),1);   %Shared edge
        M2=X0((42*(n+1)-14):(42*(n+1)-12),1);   %F2=F1 for next elem
    end

    % Substitute guess for unknowns (14n properties dependent on problem)
    q   =X0((42*n-41):(42*n-39),1);
    rho =X0((42*n-38):(42*n-36),1);
    q3  =X0((42*n-35):(42*n-33),1);
    rho3=X0((42*n-32):(42*n-30),1);
    P2  =X0((42*n-29):(42*n-27),1);
```

```
H2  =X0((42*n-26):(42*n-24),1);
q2  =X0((42*n-23):(42*n-21),1);
rho2=X0((42*n-20):(42*n-18),1);
F1  =X0((42*n-17):(42*n-15),1);
M1  =X0((42*n-14):(42*n-12),1);
V   =X0((42*n-11):(42*n -9),1);
W   =X0((42*n -8):(42*n -6),1);
G   =X0((42*n -5):(42*n -3),1);
K   =X0((42*n -2):(42*n),1)   ;

% For steady problems, displacements and momenta are constant over time
q1  =q3;  % Enforce kinematics as functions of rigid body motion
rho1=rho3;%
P1  =P2;  % Enforce momenta as constant functions of kinematics
H1  =H2;  %

%Assign intermediate variables
dx=SIMS.dx;
dt=1;     % Auto for steady analysis
mass=MATL.mass(n);
P=1/2*(P1+P2);
H=1/2*(H1+H2);
F=1/2*(F1+F2);
M=1/2*(M1+M2);
[HH,C]=Milenkovic(rho);
g=C*LOAD.G0;              %used in deformed coordinate system
r_cg=C*GEOM.r_cg_b(:,n); %used in deformed coordinate system
f=(LOAD.f0(:,n)+mass*g)*t/SIMS.num_q;
m=(LOAD.m0(:,n)+mass*tilde(r_cg)*g)*t/SIMS.num_q;
v=RIGD.v0(:,n)*t/SIMS.num_q;
w=RIGD.w0*t/SIMS.num_q;
k=GEOM.k_b(:,n);
I=MATL.I_b(:,:,n);
A=MATL.A(:,:,n);
B=MATL.B(:,:,n);
D=MATL.D(:,:,n);
e1=GEOM.e1;

%Evaluate 14 equations using guess
delq=dx/4*(P1-P2)-dx*dt/4*tilde(W)*P+dt/4*(F2-F1)+...
    dx*dt/4*tilde(K)*F+dx*dt/4*f;
delp=dx/4*(H1-H2)-dx*dt/4*tilde(W)*H-dx*dt/4*tilde(V)*...
    P+dx*dt/4*tilde(e1+G)*F+dx*dt/4*tilde(K)*M+dt/4*(M2-M1)+dx*dt/4*m;
delP1=dt/2*(v+tilde(w)*q-C'*V)+(q-q1);
delP2=dt/2*(v+tilde(w)*q-C'*V)+(q3-q);
delH1=dt/2*inv(HH)*(C*w-W)+(rho-rho1);
```

```
delH2=dt/2*inv(HH)*(C*w-W)+(rho3-rho);
delF1=dx/2*(C'*(G+e1)-(e1+tilde(k)*q))+(q4-q);
delF2=dx/2*(C'*(G+e1)-(e1+tilde(k)*q))+(q-q2);
delM1=dx/2*inv(HH)*(K-C*k)+(rho4-rho);
delM2=dx/2*inv(HH)*(K-C*k)+(rho-rho2);
delV=P-mass*V+mass*tilde(r_cg)*W;
delW=H-I*W-mass*tilde(r_cg)*V;
delG=A*G+B*(K-k)-F;
delK=B'*G+D*(K-k)-M;

%Assign residuals
residual((42*n-41):(42*n-39),1)=delq;
residual((42*n-38):(42*n-36),1)=delp;
residual((42*n-35):(42*n-33),1)=delP1;
residual((42*n-32):(42*n-30),1)=delP2;
residual((42*n-29):(42*n-27),1)=delH1;
residual((42*n-26):(42*n-24),1)=delH2;
residual((42*n-23):(42*n-21),1)=delF1;
residual((42*n-20):(42*n-18),1)=delF2;
residual((42*n-17):(42*n-15),1)=delM1;
residual((42*n-14):(42*n-12),1)=delM2;
residual((42*n-11):(42*n- 9),1)=delV;
residual((42*n- 8):(42*n- 6),1)=delW;
residual((42*n- 5):(42*n- 3),1)=delG;
residual((42*n- 2):(42*n),1)   =delK;

%Define individual vector elements used in calculating the Jacobian
r1=X0((42*n-38),1); %rotation parameter 1- from rho bar
r2=X0((42*n-37),1); %rotation parameter 2- from rho bar
r3=X0((42*n-36),1); %rotation parameter 3- from rho bar
V1=X0((42*n-11),1); %velocity 1
V2=X0((42*n-10),1); %velocity 2
V3=X0((42*n- 9),1); %velocity 3
G1=X0((42*n-5),1)+1;%strain 1 + e1
G2=X0((42*n-4),1);  %strain 2
G3=X0((42*n-3),1);  %strain 3
K1=X0((42*n-2),1);  %curvature 1
K2=X0((42*n-1),1);  %curvature 2
K3=X0((42*n),1);    %curvature 3
w1=w(1);            %prescribed angular velocity 1
w2=w(2);            %prescribed angular velocity 2
w3=w(3);            %prescribed angular velocity 3
k1=k(1);            %initial twist
k2=k(2);            %initial curvature 2
k3=k(3);            %initial curvature 3
W1=X0(42*n -8,1);   %angular velocity 1
```

```
W2=X0(42*n -7,1);   %angular velocity 2
W3=X0(42*n -6,1);   %angular velocity 3

%Define Jacobian Components from rotation parameters:

%Partial [C'V] / Partial r
JAden=(16+r1^2+r2^2+r3^2)^3;
JA11=32*(r1^3*(-r3*V2+r2*V3)-12*r1^2*(r2*V2+r3*V3)+4*(16+r2^2+r3^2)...
    *(r2*V2+r3*V3)+r1*(r2^2*(16*V1-r3*V2)+r3*(16*r3*V1+48*V2-r3^2*...
    V2)+r2^3*V3+r2*(-48+r3^2)*V3));
JA21=-16*(-8*r2^3*V1-2*r1^3*(r3*V1+8*V2)-2*r1*(-48*r3*V1+r3^3*V1-...
    128*V2+8*r3^2*V2)-96*r1^2*V3+r1^4*V3-r2^4*V3-(-256+r3^4)*V3+8*r2...
    *((-16+3*r1^2-r3^2)*V1+4*r1*r3*V3)-2*r2^2*(r1*r3*V1-8*r1*V2+...
    r3^2*V3));
JA31=16*(r1^4*V2-24*r1^2*(r3*V1+4*V2)-(16+r2^2+r3^2)*(-8*r3*V1+...
    (-16+r2^2)*V2+r3^2*V2)-2*r1^3*(r2*V1-8*V3)-2*r1*(r2^3*V1+r2*...
    ((-48+r3^2)*V1+16*r3*V2)-8*r2^2*V3+8*(16+r3^2)*V3));
JA12=-16*(2*r2^3*(-8*V1+r3*V2)-r2^4*V3+24*r2^2*(r1*V2+4*V3)+(16+...
    r1^2+r3^2)*(-8*r1*V2+r1^2*V3+(-16+r3^2)*V3)+2*r2*(8*(16+r1^2-...
    r3^2)*V1+r3*((-48+r1^2+r3^2)*V2+16*r1*V3)));
JA22=-32*(r1^3*(-4*V1+r2*V3)-r1^2*(r2*r3*V1+16*r2*V2+4*r3*V3)-r3*...
    (r2^3*V1+r2*((-48+r3^2)*V1+16*r3*V2)-12*r2^2*V3+4*(16+r3^2)*...
    V3)+r1*(4*(-16+3*r2^2-r3^2)*V1+r2*(-48+r2^2+r3^2)*V3));
JA32=16*((-256+r1^4+96*r2^2-r2^4-32*r1*r2*r3+2*r1^2*r3^2+r3^4)*V1+...
    2*(r1^3*r2*V2+r1*r2*(-48+r2^2+r3^2)*V2+4*r1^2*(r3*V2+2*r2*V3)+...
    4*((16-3*r2^2)*r3*V2+r3^2*V2+2*r2*(-16+r2^2)*V3-2*r2*r3^2*V3)));
JA13=16*(-r3^4*V2+24*r3^2*(4*V2-r1*V3)+(16+r1^2+r2^2)*((-16+r1^2+...
    r2^2)*V2+8*r1*V3)+2*r3^3*(8*V1+r2*V3)-2*r3*(8*(16+r1^2-r2^2)*...
    V1-r2*(-16*r1*V2+r1^2*V3+(-48+r2^2)*V3)));
JA23=-16*((-256+r1^4+2*r1^2*r2^2+r2^4+32*r1*r2*r3+96*r3^2-r3^4)*...
    V1+2*(-4*r2*(16+r1^2+r2^2)*V3+12*r2*r3^2*V3+r3^3*(-8*V2+r1*...
    V3)+r3*(-8*(-16+r1^2-r2^2)*V2+r1*(-48+r1^2+r2^2)*V3)));
JA33=32*(r1^3*(4*V1+r3*V2)+r1*(4*(16+r2^2-3*r3^2)*V1+r3*(-48+...
    r2^2+r3^2)*V2)+r1^2*(-r2*r3*V1+4*r2*V2+16*r3*V3)+r2*(-r3^3*...
    V1+4*(16+r2^2)*V2-12*r3^2*V2+r3*(48*V1-r2^2*V1+16*r2*V3)));
JA=1/JAden*[JA11 JA12 JA13;
            JA21 JA22 JA23;
            JA31 JA32 JA33];

%Partial [(H')^-1 w] / Partial r
JB11=r1*w1+r2*w2+r3*w3;
JB21=r2*w1-r1*w2+4*w3;
JB31=r3*w1-4*w2-r1*w3;
JB12=-r2*w1+r1*w2-4*w3;
JB22=r1*w1+r2*w2+r3*w3;
JB32=4*w1+r3*w2-r2*w3;
```

```
JB13=-r3*w1+4*w2+r1*w3;
JB23=-4*w1-r3*w2+r2*w3;
JB33=r1*w1+r2*w2+r3*w3;
JB=1/8*[JB11 JB12 JB13;
        JB21 JB22 JB23;
        JB31 JB32 JB33];

%Partial [(H)^-1 W] / Partial r
JC11=r1*W1+r2*W2+r3*W3;
JC21=-4*W3-r1*W2+r2*W1;
JC31=4*W2-r1*W3+r3*W1;
JC12=4*W3-r1*W2-r2*W1;
JC22=r1*W1+r2*W2+r3*W3;
JC32=-4*W1-r2*W3+r3*W2;
JC13=-4*W2+r1*W3-r3*W1;
JC23=4*W1+r2*W3-r3*W2;
JC33=r1*W1+r2*W2+r3*W3;
JC=1/8*[JC11 JC12 JC13;
        JC21 JC22 JC23;
        JC31 JC32 JC33];

%Partial [C'(G+e1)] / Partial r
JDden=(16+r1^2+r2^2+r3^2)^3;
JD11=32*(r1^3*(-r3*G2+r2*G3)-12*r1^2*(r2*G2+r3*G3)+4*(16+r2^2+...
    r3^2)*(r2*G2+r3*G3)+r1*(r2^2*(16*G1-r3*G2)+r3*(16*r3*G1+48*...
    G2-r3^2*G2)+r2^3*G3+r2*(-48+r3^2)*G3));
JD21=-16*(-8*r2^3*G1-2*r1^3*(r3*G1+8*G2)-2*r1*(-48*r3*G1+r3^3*...
    G1-128*G2+8*r3^2*G2)-96*r1^2*G3+r1^4*G3-r2^4*G3-(-256+r3^4)*...
    G3+8*r2*((-16+3*r1^2-r3^2)*G1+4*r1*r3*G3)-2*r2^2*(r1*r3*G1-8*...
    r1*G2+r3^2*G3));
JD31=16*(r1^4*G2-24*r1^2*(r3*G1+4*G2)-(16+r2^2+r3^2)*(-8*r3*G1+...
    (-16+r2^2)*G2+r3^2*G2)-2*r1^3*(r2*G1-8*G3)-2*r1*(r2^3*G1+r2*...
    ((-48+r3^2)*G1+16*r3*G2)-8*r2^2*G3+8*(16+r3^2)*G3));
JD12=-16*(2*r2^3*(-8*G1+r3*G2)-r2^4*G3+24*r2^2*(r1*G2+4*G3)+(16+...
    r1^2+r3^2)*(-8*r1*G2+r1^2*G3+(-16+r3^2)*G3)+2*r2*(8*(16+r1^2-...
    r3^2)*G1+r3*((-48+r1^2+r3^2)*G2+16*r1*G3)));
JD22=-32*(r1^3*(-4*G1+r2*G3)-r1^2*(r2*r3*G1+16*r2*G2+4*r3*G3)-r3*...
    (r2^3*G1+r2*((-48+r3^2)*G1+16*r3*G2)-12*r2^2*G3+4*(16+r3^2)*...
    G3)+r1*(4*(-16+3*r2^2-r3^2)*G1+r2*(-48+r2^2+r3^2)*G3));
JD32=16*((-256+r1^4+96*r2^2-r2^4-32*r1*r2*r3+2*r1^2*r3^2+r3^4)*...
    G1+2*(r1^3*r2*G2+r1*r2*(-48+r2^2+r3^2)*G2+4*r1^2*(r3*G2+2*r2*...
    G3)+4*((16-3*r2^2)*r3*G2+r3^2*G2+2*r2*(-16+r2^2)*G3-2*r2*r3^2*...
    G3)));
JD13=16*(-r3^4*G2+24*r3^2*(4*G2-r1*G3)+(16+r1^2+r2^2)*((-16+...
    r1^2+r2^2)*G2+8*r1*G3)+2*r3^3*(8*G1+r2*G3)-2*r3*(8*(16+...
    r1^2-r2^2)*G1-r2*(-16*r1*G2+r1^2*G3+(-48+r2^2)*G3)));
```

```
JD23=-16*((-256+r1^4+2*r1^2*r2^2+r2^4+32*r1*r2*r3+96*r3^2-r3^4)*...
    G1+2*(-4*r2*(16+r1^2+r2^2)*G3+12*r2*r3^2*G3+r3^3*(-8*G2+r1*...
    G3)+r3*(-8*(-16+r1^2-r2^2)*G2+r1*(-48+r1^2+r2^2)*G3)));
JD33=32*(r1^3*(4*G1+r3*G2)+r1*(4*(16+r2^2-3*r3^2)*G1+r3*(-48+...
    r2^2+r3^2)*G2)+r1^2*(-r2*r3*G1+4*r2*G2+16*r3*G3)+r2*(-r3^3*G1+...
    4*(16+r2^2)*G2-12*r3^2*G2+r3*(48*G1-r2^2*G1+16*r2*G3)));
JD=1/JDden*[JD11 JD12 JD13;
            JD21 JD22 JD23;
            JD31 JD32 JD33];

%Partial [(H')^-1 k] / Partial r
JE11=r1*k1+r2*k2+r3*k3;
JE21=r2*k1-r1*k2+4*k3;
JE31=r3*k1-4*k2-r1*k3;
JE12=-r2*k1+r1*k2-4*k3;
JE22=r1*k1+r2*k2+r3*k3;
JE32=4*k1+r3*k2-r2*k3;
JE13=-r3*k1+4*k2+r1*k3;
JE23=-4*k1-r3*k2+r2*k3;
JE33=r1*k1+r2*k2+r3*k3;
JE=1/8*[JE11 JE12 JE13;
        JE21 JE22 JE23;
        JE31 JE32 JE33];

%Partial [(H)^-1 K] / Partial r
JF11=r1*K1+r2*K2+r3*K3;
JF21=-4*K3-r1*K2+r2*K1;
JF31=4*K2-r1*K3+r3*K1;
JF12=4*K3-r1*K2-r2*K1;
JF22=r1*K1+r2*K2+r3*K3;
JF32=-4*K1-r2*K3+r3*K2;
JF13=-4*K2+r1*K3-r3*K1;
JF23=4*K1+r2*K3-r3*K2;
JF33=r1*K1+r2*K2+r3*K3;
JF=1/8*[JF11 JF12 JF13;
        JF21 JF22 JF23;
        JF31 JF32 JF33];

%Fill the n x n Jacobian block with non zero elements (diagonal block)
Jacobian((42*n-35):(42*n-33),(42*n-41):(42*n-39))=...
    dt/2*tilde(w)+eye(3);%delP1/q
Jacobian((42*n-32):(42*n-30),(42*n-41):(42*n-39))=...
    dt/2*tilde(w)-eye(3);%delP2/q
Jacobian((42*n-23):(42*n-21),(42*n-41):(42*n-39))=...
    -dx/2*tilde(k)-eye(3);%delF1/q
Jacobian((42*n-20):(42*n-18),(42*n-41):(42*n-39))=...
```

```
    -dx/2*tilde(k)+eye(3);%delF2/q
Jacobian((42*n-35):(42*n-33),(42*n-38):(42*n-36))=-dt/2*JA;%delP1/rho
Jacobian((42*n-32):(42*n-30),(42*n-38):(42*n-36))=-dt/2*JA;%delP2/rho
Jacobian((42*n-29):(42*n-27),(42*n-38):(42*n-36))=...
    dt/2*(JB-JC)+eye(3);%delH1/rho
Jacobian((42*n-26):(42*n-24),(42*n-38):(42*n-36))=...
    dt/2*(JB-JC)-eye(3);%delH2/rho
Jacobian((42*n-23):(42*n-21),(42*n-38):(42*n-36))=dx/2*JD;%delF1/rho
Jacobian((42*n-20):(42*n-18),(42*n-38):(42*n-36))=dx/2*JD;%delF2/rho
Jacobian((42*n-17):(42*n-15),(42*n-38):(42*n-36))=...
    dx/2*(JF-JE)-eye(3);%delM1/rho
Jacobian((42*n-14):(42*n-12),(42*n-38):(42*n-36))=...
    dx/2*(JF-JE)+eye(3);%delM2/rho
Jacobian((42*n-35):(42*n-33),(42*n-35):(42*n-33))=...
    -eye(3);%delP1/q3-only for steady problems
Jacobian((42*n-32):(42*n-30),(42*n-35):(42*n-33))=eye(3);%delP2/q3
Jacobian((42*n-29):(42*n-27),(42*n-32):(42*n-30))=...
    -eye(3);%delH1/rho3-only for steady problems
Jacobian((42*n-26):(42*n-24),(42*n-32):(42*n-30))=eye(3);%delH2/rho3
Jacobian((42*n-41):(42*n-39),(42*n-29):(42*n-27))=...
    -dx*dt/4*tilde(W);%delq/P2-steady
Jacobian((42*n-38):(42*n-36),(42*n-29):(42*n-27))=...
    -dx*dt/4*tilde(V);%delpsi/P2
Jacobian((42*n-11):(42*n-9),(42*n-29):(42*n-27))=eye(3);%delV/P2-steady
Jacobian((42*n-38):(42*n-36),(42*n-26):(42*n-24))=...
    -dx*dt/4*tilde(W);%delpsi/H2-steady
Jacobian((42*n-8):(42*n-6),(42*n-26):(42*n-24))=eye(3);%delW/H2-steady
Jacobian((42*n-20):(42*n-18),(42*n-23):(42*n-21))=-eye(3);%delF2/q2
Jacobian((42*n-14):(42*n-12),(42*n-20):(42*n-18))=-eye(3);%delM2/rho2
Jacobian((42*n-41):(42*n-39),(42*n-17):(42*n-15))=...
    -dt/4*eye(3)+dx*dt/8*tilde(K);%delq/F1
Jacobian((42*n-38):(42*n-36),(42*n-17):(42*n-15))=...
    dx*dt/8*tilde(e1+G);%delpsi/F1
Jacobian((42*n-5):(42*n-3),(42*n-17):(42*n-15))=-1/2*eye(3);%delG/F1
Jacobian((42*n-38):(42*n-36),(42*n-14):(42*n-12))=...
    -dt/4*eye(3)+dx*dt/8*tilde(K);%delpsi/M1
Jacobian((42*n-2):(42*n),(42*n-14):(42*n-12))=-1/2*eye(3);%delK/M1
Jacobian((42*n-38):(42*n-36),(42*n-11):(42*n-9))=...
    dx*dt/8*tilde(P1+P2);%delpsi/V
Jacobian((42*n-35):(42*n-33),(42*n-11):(42*n-9))=-dt/2*C';%delP1/V
Jacobian((42*n-32):(42*n-30),(42*n-11):(42*n-9))=-dt/2*C';%delP2/V
Jacobian((42*n-11):(42*n-9),(42*n-11):(42*n-9))=-mass*eye(3);%delV/V
Jacobian((42*n-8):(42*n-6),(42*n-11):(42*n-9))=-mass*tilde(r_cg);%delW/V
Jacobian((42*n-41):(42*n-39),(42*n-8):(42*n-6))=...
    dx*dt/8*tilde(P1+P2);%delq/W
Jacobian((42*n-38):(42*n-36),(42*n-8):(42*n-6))=...
```

```
        dx*dt/8*tilde(H1+H2);%delpsi/W
    Jacobian((42*n-29):(42*n-27),(42*n-8):(42*n-6))=-dt/2*inv(HH);%delH1/W
    Jacobian((42*n-26):(42*n-24),(42*n-8):(42*n-6))=-dt/2*inv(HH);%delH2/W
    Jacobian((42*n-11):(42*n-9),(42*n-8):(42*n-6))=mass*tilde(r_cg);%delV/W
    Jacobian((42*n-8):(42*n-6),(42*n-8):(42*n-6))=-I;%delW/W
    Jacobian((42*n-38):(42*n-36),(42*n-5):(42*n-3))=...
        -dx*dt/8*tilde(F1+F2);%delpsi/G
    Jacobian((42*n-23):(42*n-21),(42*n-5):(42*n-3))=dx/2*C';%delF1/G
    Jacobian((42*n-20):(42*n-18),(42*n-5):(42*n-3))=dx/2*C';%delF2/G
    Jacobian((42*n-5):(42*n-3),(42*n-5):(42*n-3))=A;%delG/G
    Jacobian((42*n-2):(42*n),(42*n-5):(42*n-3))=B';%delK/G
    Jacobian((42*n-41):(42*n-39),(42*n-2):(42*n))=...
        -dx*dt/8*tilde(F1+F2);%delq/K
    Jacobian((42*n-38):(42*n-36),(42*n-2):(42*n))=...
        -dx*dt/8*tilde(M1+M2);%delpsi/K
    Jacobian((42*n-17):(42*n-15),(42*n-2):(42*n))=dx/2*inv(HH);%delM1/K
    Jacobian((42*n-14):(42*n-12),(42*n-2):(42*n))=dx/2*inv(HH);%delM2/K
    Jacobian((42*n-5):(42*n-3),(42*n-2):(42*n))=B;%delG/K
    Jacobian((42*n-2):(42*n),(42*n-2):(42*n))=D;%delK/K

    %Contributions to off diagonal blocks
    if n > 1
        %The n x (n-1) block
        Jacobian((42*n-23):(42*n-21),(42*(n-1)-23):(42*(n-1)-21))=...
            eye(3);%delF1(n)/q2(n-1),ie q4
        Jacobian((42*n-17):(42*n-15),(42*(n-1)-20):(42*(n-1)-18))=...
            eye(3);%delM1(n)/rho2(n-1),ie rho4
    end

    if n < SIMS.num_x
        %The n x (n+1) block
        Jacobian((42*n-41):(42*n-39),(42*(n+1)-17):(42*(n+1)-15))=...
            dt/4*eye(3)+dx*dt/8*tilde(K);%delq(n)/F(n+1), ie F2
        Jacobian((42*n-38):(42*n-36),(42*(n+1)-17):(42*(n+1)-15))=...
            dx*dt/8*tilde(e1+G);%delpsi(n)/F(n+1)
        Jacobian((42*n-5):(42*n-3),(42*(n+1)-17):(42*(n+1)-15))=...
            -1/2*eye(3);%delG(n)/F(n+1)
        Jacobian((42*n-38):(42*n-36),(42*(n+1)-14):(42*(n+1)-12))=...
            dt/4*eye(3)+dx*dt/8*tilde(K);%delpsi(n)/M(n+1), ie M2
        Jacobian((42*n-2):(42*n),(42*(n+1)-14):(42*(n+1)-12))=...
            -1/2*eye(3);%delG(n)/M(n+1)
    end

end
%%-------------------------------------------------------------------------%%
```

D.8 FEDynamic.m

The following routine is used to solve dynamic problems. After the complete initial conditions are obtained using one of the two steady-state algorithms, they are passed to this file. It solves the system of equations for each time step, updating the initial conditions after each iteration.

```
%%---------------------------------------------------------------------%%
% Obtain Space-time FE Solution

t_end=SIMS.dt;      % time stamp at end of interval
t_mid=SIMS.dt/2;    % time stamp at middle of interval
sample=0;           % output counter
tag=0;              % output index

for t=1:SIMS.num_t;
    % Solve individual time slab
    Options=optimset('Display','off','MaxIter',40000,'TolFun',...
        SIMS.dx*SIMS.dt*1e-10,'TolX',SIMS.dx*SIMS.dt*1e-10,...
        'MaxFunEvals',40000,'NonlEqnAlgorithm','gn','Jacobian','on');
   [X,FVAL(t,:),EXITFLAG(t),OUTPUT(t),JACOB]=fsolve(@Dynamic,X0,Options,t);

    % Store simulation data and time stamps at specified sampling intervals
    sample=sample+1;
    if sample == SIMS.sample;
        tag=tag+1;
        XDynamic(1:42*SIMS.num_x,tag)=X;
        XDynamic(42*SIMS.num_x+1,tag)=t_mid;
        XDynamic(42*SIMS.num_x+2,tag)=t_end;
        sample=0;
        marker=sprintf('Completed iteration %d',t);
        disp(marker);
    end

    % Use numerical derivatives to project guess for next time step
    if t==1;     %half-step first derivative
        X0=3*X-2*XStatic;
        Xsub1=X;
    elseif t==2; %first derivative and half-step second derivative
        X0=3*X-4*Xsub1+2*XStatic;
        Xsub2=Xsub1;
        Xsub1=X;
    else         %first and second derivatives
        X0=3*X-3*Xsub1+Xsub2;
        Xsub2=Xsub1;
```

```
        Xsub1=X;
    end

    % Assign initial conditions for the next time step
    if t < SIMS.num_t,
        for n=1:SIMS.num_x;
            IC.q1_b(:,n,t+1)=X((42*n-35):(42*n-33),1);
            IC.r1_b(:,n,t+1)=X((42*n-32):(42*n-30),1);
            IC.P1_B(:,n,t+1)=X((42*n-29):(42*n-27),1);
            IC.H1_B(:,n,t+1)=X((42*n-26):(42*n-24),1);
        end
        % Update clock
        t_end=t_end+SIMS.dt;
        t_mid=t_mid+SIMS.dt;
    end
end
% Print to file
dlmwrite(DName,XDynamic,'\t');
%%-------------------------------------------------------------------------%%
```

D.9 Dynamic.m

This function is used by fsolve.m to evaluate the residuals and Jacobian for dynamic problems. There are some differences in the Jacobian where quantities are no longer constant with respect to time. Likewise, the constraints from equation (3.106) are removed and replaced with known initial conditions.

```
%%-------------------------------------------------------------------------%%
function [residual,Jacobian]=Dynamic(X0,t)
% Minimizes residuals of 14n equations inside a single time step

global GEOM  % geometric properties
global MATL  % material properties
global RIGD  % rigid body motion
global LOAD  % distributed applied loads
global SIMS  % simulation parameters
global BC    % boundary conditions
global IC    % initial conditions

%Initialize Jacobian
Jacobian=zeros(42*SIMS.num_x,42*SIMS.num_x);

% Calculate residuals and Jacobian using X0 as the guess for the unknowns
for n=1:SIMS.num_x;
```

```
% Boundary conditions and shared edges (problem dependent)
if n == 1,
    q4=BC.q4_b(:,1,t);                      %Boundary condition
    rho4=BC.r4_b(:,1,t);                    % " "
else
    q4=X0((42*(n-1)-23):(42*(n-1)-21),1);   %Shared edge
    rho4=X0((42*(n-1)-20):(42*(n-1)-18),1); %q4=q2 for prev elem
end

if n == SIMS.num_x,
    F2=BC.F2_B(:,SIMS.num_x,t);           %Boundary condition
    M2=BC.M2_B(:,SIMS.num_x,t);           % " "
else
    F2=X0((42*(n+1)-17):(42*(n+1)-15),1); %Shared edge
    M2=X0((42*(n+1)-14):(42*(n+1)-12),1); %F2=F1 for next elem
end

%Known initial conditions
P1=IC.P1_B(:,n,t);              %Initial conditions
H1=IC.H1_B(:,n,t);              % " "
q1=IC.q1_b(:,n,t);              % " "
rho1=IC.r1_b(:,n,t);            % " "

% Substitute guess for unknowns (14n properties dependent on problem)
q   =X0((42*n-41):(42*n-39),1);
rho =X0((42*n-38):(42*n-36),1);
q3  =X0((42*n-35):(42*n-33),1);
rho3=X0((42*n-32):(42*n-30),1);
P2  =X0((42*n-29):(42*n-27),1);
H2  =X0((42*n-26):(42*n-24),1);
q2  =X0((42*n-23):(42*n-21),1);
rho2=X0((42*n-20):(42*n-18),1);
F1  =X0((42*n-17):(42*n-15),1);
M1  =X0((42*n-14):(42*n-12),1);
V   =X0((42*n-11):(42*n -9),1);
W   =X0((42*n -8):(42*n -6),1);
G   =X0((42*n -5):(42*n -3),1);
K   =X0((42*n -2):(42*n),1)   ;

%Assign intermediate variables
dx=SIMS.dx;
dt=SIMS.dt;
mass=MATL.mass(n);
P=1/2*(P1+P2);
H=1/2*(H1+H2);
```

```
F=1/2*(F1+F2);
M=1/2*(M1+M2);
[HH,C]=Milenkovic(rho);
g=C*LOAD.G(:,t);         %used in deformed coordinate system
r_cg=C*GEOM.r_cg_b(:,n); %used in deformed coordinate system
v=RIGD.v_b(:,n,t);
w=RIGD.w_b(:,t);
k=GEOM.k_b(:,n);
I=MATL.I_b(:,:,n);
A=MATL.A(:,:,n);
B=MATL.B(:,:,n);
D=MATL.D(:,:,n);
e1=GEOM.e1;
f=LOAD.f_B(:,n,t)+mass*g-SIMS.edf*(V-C*(v+tilde(w)*q));
m=LOAD.m_B(:,n,t)+mass*tilde(r_cg)*g-SIMS.edm*(W-C*w);

% Evaluate 14 equations using guess
delq=dx/4*(P1-P2)-dx*dt/4*tilde(W)*P+dt/4*(F2-F1)+dx*dt/4*...
    tilde(K)*F+dx*dt/4*f;
delp=dx/4*(H1-H2)-dx*dt/4*tilde(W)*H-dx*dt/4*tilde(V)*P+...
    dx*dt/4*tilde(e1+G)*F+dx*dt/4*tilde(K)*M+dt/4*(M2-M1)+dx*dt/4*m;
delP1=dt/2*(v+tilde(w)*q-C'*V)+(q-q1);
delP2=dt/2*(v+tilde(w)*q-C'*V)+(q3-q);
delH1=dt/2*inv(HH)*(C*w-W)+(rho-rho1);
delH2=dt/2*inv(HH)*(C*w-W)+(rho3-rho);
delF1=dx/2*(C'*(G+e1)-(e1+tilde(k)*q))+(q4-q);
delF2=dx/2*(C'*(G+e1)-(e1+tilde(k)*q))+(q-q2);
delM1=dx/2*inv(HH)*(K-C*k)+(rho4-rho);
delM2=dx/2*inv(HH)*(K-C*k)+(rho-rho2);
delV=P-mass*V+mass*tilde(r_cg)*W;
delW=H-I*W-mass*tilde(r_cg)*V;
delG=A*G+B*(K-k)-F;
delK=B'*G+D*(K-k)-M;

%Assign residuals
residual((42*n-41):(42*n-39),1)=delq;
residual((42*n-38):(42*n-36),1)=delp;
residual((42*n-35):(42*n-33),1)=delP1;
residual((42*n-32):(42*n-30),1)=delP2;
residual((42*n-29):(42*n-27),1)=delH1;
residual((42*n-26):(42*n-24),1)=delH2;
residual((42*n-23):(42*n-21),1)=delF1;
residual((42*n-20):(42*n-18),1)=delF2;
residual((42*n-17):(42*n-15),1)=delM1;
residual((42*n-14):(42*n-12),1)=delM2;
residual((42*n-11):(42*n- 9),1)=delV;
```

```
residual((42*n- 8):(42*n- 6),1)=delW;
residual((42*n- 5):(42*n- 3),1)=delG;
residual((42*n- 2):(42*n),1)   =delK;

%Define individual vector elements used in calculating the Jacobian
r1=X0((42*n-38),1); %rotation parameter 1- from rho bar
r2=X0((42*n-37),1); %rotation parameter 2- from rho bar
r3=X0((42*n-36),1); %rotation parameter 3- from rho bar
V1=X0((42*n-11),1); %velocity 1
V2=X0((42*n-10),1); %velocity 2
V3=X0((42*n- 9),1); %velocity 3
G1=X0((42*n-5),1)+1;%strain 1(includes e1)
G2=X0((42*n-4),1);  %strain 2
G3=X0((42*n-3),1);  %strain 3
K1=X0((42*n-2),1);  %curvature 1
K2=X0((42*n-1),1);  %curvature 2
K3=X0((42*n),1);    %curvature 3
w1=w(1);            %prescribed angular velocity 1
w2=w(2);            %prescribed angular velocity 2
w3=w(3);            %prescribed angular velocity 3
k1=k(1);            %initial twist
k2=k(2);            %initial curvature 2
k3=k(3);            %initial curvature 3
W1=X0(42*n -8,1);   %angular velocity 1
W2=X0(42*n -7,1);   %angular velocity 2
W3=X0(42*n -6,1);   %angular velocity 3
Vr=v+tilde(w)*q;    %velocity not due to strain rates
Vr1=Vr(1);          % " " 1
Vr2=Vr(2);          % " " 2
Vr3=Vr(3);          % " " 3

%Assemble Jacobian Components for rotation matrices

%Partial [C'V] / Partial r
JAden=(16+r1^2+r2^2+r3^2)^3;
JA11=32*(r1^3*(-r3*V2+r2*V3)-12*r1^2*(r2*V2+r3*V3)+4*(16+r2^2+r3^2)...
   *(r2*V2+r3*V3)+r1*(r2^2*(16*V1-r3*V2)+r3*(16*r3*V1+48*V2-r3^2*V2)...
    +r2^3*V3+r2*(-48+r3^2)*V3));
JA21=-16*(-8*r2^3*V1-2*r1^3*(r3*V1+8*V2)-2*r1*(-48*r3*V1+r3^3*V1-...
    128*V2+8*r3^2*V2)-96*r1^2*V3+r1^4*V3-r2^4*V3-(-256+r3^4)*V3+...
    8*r2*((-16+3*r1^2-r3^2)*V1+4*r1*r3*V3)-2*r2^2*...
    (r1*r3*V1-8*r1*V2+r3^2*V3));
JA31=16*(r1^4*V2-24*r1^2*(r3*V1+4*V2)-(16+r2^2+r3^2)*(-8*r3*V1+...
    (-16+r2^2)*V2+r3^2*V2)-2*r1^3*(r2*V1-8*V3)-2*r1*(r2^3*V1+r2*...
    ((-48+r3^2)*V1+16*r3*V2)-8*r2^2*V3+8*(16+r3^2)*V3));
JA12=-16*(2*r2^3*(-8*V1+r3*V2)-r2^4*V3+24*r2^2*(r1*V2+4*V3)+...
```

```
    (16+r1^2+r3^2)*(-8*r1*V2+r1^2*V3+(-16+r3^2)*V3)+2*r2*(8*...
    (16+r1^2-r3^2)*V1+r3*((-48+r1^2+r3^2)*V2+16*r1*V3)));
JA22=-32*(r1^3*(-4*V1+r2*V3)-r1^2*(r2*r3*V1+16*r2*V2+4*r3*V3)-...
    r3*(r2^3*V1+r2*((-48+r3^2)*V1+16*r3*V2)-12*r2^2*V3+4*(16+r3^2)...
    *V3)+r1*(4*(-16+3*r2^2-r3^2)*V1+r2*(-48+r2^2+r3^2)*V3));
JA32=16*((-256+r1^4+96*r2^2-r2^4-32*r1*r2*r3+2*r1^2*r3^2+r3^4)*V1+...
    2*(r1^3*r2*V2+r1*r2*(-48+r2^2+r3^2)*V2+4*r1^2*(r3*V2+2*r2*V3)+...
    4*((16-3*r2^2)*r3*V2+r3^3*V2+2*r2*(-16+r2^2)*V3-2*r2*r3^2*V3)));
JA13=16*(-r3^4*V2+24*r3^2*(4*V2-r1*V3)+(16+r1^2+r2^2)*((-16+r1^2+...
    r2^2)*V2+8*r1*V3)+2*r3^3*(8*V1+r2*V3)-2*r3*(8*(16+r1^2-r2^2)*V1...
    -r2*(-16*r1*V2+r1^2*V3+(-48+r2^2)*V3)));
JA23=-16*((-256+r1^4+2*r1^2*r2^2+r2^4+32*r1*r2*r3+96*r3^2-r3^4)*...
    V1+2*(-4*r2*(16+r1^2+r2^2)*V3+12*r2*r3^2*V3+r3^3*(-8*V2+r1*V3)+...
    r3*(-8*(-16+r1^2-r2^2)*V2+r1*(-48+r1^2+r2^2)*V3)));
JA33=32*(r1^3*(4*V1+r3*V2)+r1*(4*(16+r2^2-3*r3^2)*V1+r3*(-48+r2^2+...
    r3^2)*V2)+r1^2*(-r2*r3*V1+4*r2*V2+16*r3*V3)+r2*(-r3^3*V1+4*(16+...
    r2^2)*V2-12*r3^2*V2+r3*(48*V1-r2^2*V1+16*r2*V3)));
JA=1/JAden*[JA11 JA12 JA13;
            JA21 JA22 JA23;
            JA31 JA32 JA33];

%Partial [(H')^-1 w] / Partial r
JB11=r1*w1+r2*w2+r3*w3;
JB21=r2*w1-r1*w2+4*w3;
JB31=r3*w1-4*w2-r1*w3;
JB12=-r2*w1+r1*w2-4*w3;
JB22=r1*w1+r2*w2+r3*w3;
JB32=4*w1+r3*w2-r2*w3;
JB13=-r3*w1+4*w2+r1*w3;
JB23=-4*w1-r3*w2+r2*w3;
JB33=r1*w1+r2*w2+r3*w3;
JB=1/8*[JB11 JB12 JB13;
        JB21 JB22 JB23;
        JB31 JB32 JB33];

%Partial [(H)^-1 W] / Partial r
JC11=r1*W1+r2*W2+r3*W3;
JC21=-4*W3-r1*W2+r2*W1;
JC31=4*W2-r1*W3+r3*W1;
JC12=4*W3-r1*W2-r2*W1;
JC22=r1*W1+r2*W2+r3*W3;
JC32=-4*W1-r2*W3+r3*W2;
JC13=-4*W2+r1*W3-r3*W1;
JC23=4*W1+r2*W3-r3*W2;
JC33=r1*W1+r2*W2+r3*W3;
JC=1/8*[JC11 JC12 JC13;
```

```
        JC21 JC22 JC23;
        JC31 JC32 JC33];

%Partial [C'(G+e1)] / Partial r
JDden=(16+r1^2+r2^2+r3^2)^3;
JD11=32*(r1^3*(-r3*G2+r2*G3)-12*r1^2*(r2*G2+r3*G3)+4*(16+r2^2+r3^2)...
    *(r2*G2+r3*G3)+r1*(r2^2*(16*G1-r3*G2)+r3*(16*r3*G1+48*G2-r3^2*...
    G2)+r2^3*G3+r2*(-48+r3^2)*G3));
JD21=-16*(-8*r2^3*G1-2*r1^3*(r3*G1+8*G2)-2*r1*(-48*r3*G1+r3^3*G1-...
    128*G2+8*r3^2*G2)-96*r1^2*G3+r1^4*G3-r2^4*G3-(-256+r3^4)*G3+8*...
    r2*((-16+3*r1^2-r3^2)*G1+4*r1*r3*G3)-2*r2^2*(r1*r3*G1-8*r1*G2+...
    r3^2*G3));
JD31=16*(r1^4*G2-24*r1^2*(r3*G1+4*G2)-(16+r2^2+r3^2)*(-8*r3*G1+(-16+...
    r2^2)*G2+r3^2*G2)-2*r1^3*(r2*G1-8*G3)-2*r1*(r2^3*G1+r2*((-48+...
    r3^2)*G1+16*r3*G2)-8*r2^2*G3+8*(16+r3^2)*G3));
JD12=-16*(2*r2^3*(-8*G1+r3*G2)-r2^4*G3+24*r2^2*(r1*G2+4*G3)+(16+...
    r1^2+r3^2)*(-8*r1*G2+r1^2*G3+(-16+r3^2)*G3)+2*r2*(8*(16+r1^2-...
    r3^2)*G1+r3*((-48+r1^2+r3^2)*G2+16*r1*G3)));
JD22=-32*(r1^3*(-4*G1+r2*G3)-r1^2*(r2*r3*G1+16*r2*G2+4*r3*G3)-...
    r3*(r2^3*G1+r2*((-48+r3^2)*G1+16*r3*G2)-12*r2^2*G3+4*(16+r3^2)...
    *G3)+r1*(4*(-16+3*r2^2-r3^2)*G1+r2*(-48+r2^2+r3^2)*G3));
JD32=16*((-256+r1^4+96*r2^2-r2^4-32*r1*r2*r3+2*r1^2*r3^2+r3^4)*G1...
    +2*(r1^3*r2*G2+r1*r2*(-48+r2^2+r3^2)*G2+4*r1^2*(r3*G2+2*r2*G3)...
    +4*((16-3*r2^2)*r3*G2+r3^3*G2+2*r2*(-16+r2^2)*G3-2*r2*r3^2*G3)));
JD13=16*(-r3^4*G2+24*r3^2*(4*G2-r1*G3)+(16+r1^2+r2^2)*((-16+r1^2+...
    r2^2)*G2+8*r1*G3)+2*r3^3*(8*G1+r2*G3)-2*r3*(8*(16+r1^2-r2^2)*G1...
    -r2*(-16*r1*G2+r1^2*G3+(-48+r2^2)*G3)));
JD23=-16*((-256+r1^4+2*r1^2*r2^2+r2^4+32*r1*r2*r3+96*r3^2-r3^4)*G1...
    +2*(-4*r2*(16+r1^2+r2^2)*G3+12*r2*r3^2*G3+r3^3*(-8*G2+r1*G3)+...
    r3*(-8*(-16+r1^2-r2^2)*G2+r1*(-48+r1^2+r2^2)*G3)));
JD33=32*(r1^3*(4*G1+r3*G2)+r1*(4*(16+r2^2-3*r3^2)*G1+r3*(-48+...
    r2^2+r3^2)*G2)+r1^2*(-r2*r3*G1+4*r2*G2+16*r3*G3)+r2*(-r3^3*...
    G1+4*(16+r2^2)*G2-12*r3^2*G2+r3*(48*G1-r2^2*G1+16*r2*G3)));
JD=1/JDden*[JD11 JD12 JD13;
            JD21 JD22 JD23;
            JD31 JD32 JD33];

%Partial [(H')^-1 k] / Partial r
JE11=r1*k1+r2*k2+r3*k3;
JE21=r2*k1-r1*k2+4*k3;
JE31=r3*k1-4*k2-r1*k3;
JE12=-r2*k1+r1*k2-4*k3;
JE22=r1*k1+r2*k2+r3*k3;
JE32=4*k1+r3*k2-r2*k3;
JE13=-r3*k1+4*k2+r1*k3;
JE23=-4*k1-r3*k2+r2*k3;
```

```
JE33=r1*k1+r2*k2+r3*k3;
JE=1/8*[JE11 JE12 JE13;
        JE21 JE22 JE23;
        JE31 JE32 JE33];

%Partial [(H)^-1 K] / Partial r
JF11=r1*K1+r2*K2+r3*K3;
JF21=-4*K3-r1*K2+r2*K1;
JF31=4*K2-r1*K3+r3*K1;
JF12=4*K3-r1*K2-r2*K1;
JF22=r1*K1+r2*K2+r3*K3;
JF32=-4*K1-r2*K3+r3*K2;
JF13=-4*K2+r1*K3-r3*K1;
JF23=4*K1+r2*K3-r3*K2;
JF33=r1*K1+r2*K2+r3*K3;
JF=1/8*[JF11 JF12 JF13;
        JF21 JF22 JF23;
        JF31 JF32 JF33];

%Partial [C Vr]/ Partial r - damping terms
JGden=(16+r1^2+r2^2+r3^2)^3;
JG11=-32*(r1^3*(-r3*Vr2+r2*Vr3)+12*r1^2*(r2*Vr2+r3*Vr3)-4*(16+...
    r2^2+r3^2)*(r2*Vr2+r3*Vr3)+r1*(-r2^2*(16*Vr1+r3*Vr2)-r3*...
    (16*r3*Vr1-48*Vr2+r3^2*Vr2)+r2^3*Vr3+r2*(-48+r3^2)*Vr3));
JG21=16*(8*r2^3*Vr1-2*r1^3*(r3*Vr1-8*Vr2)-2*r1*(-48*r3*Vr1+r3^3*...
    Vr1+128*Vr2-8*r3^2*Vr2)-96*r1^2*Vr3+r1^4*Vr3-r2^4*Vr3-(-256+...
    r3^4)*Vr3-8*r2*((-16+3*r1^2-r3^2)*Vr1+4*r1*r3*Vr3)-2*r2^2*(r1*...
    r3*Vr1+8*r1*Vr2+r3^2*Vr3));
JG31=-16*(r1^4*Vr2+24*r1^2*(r3*Vr1-4*Vr2)-(16+r2^2+r3^2)*(8*r3*...
    Vr1+(-16+r2^2)*Vr2+r3^2*Vr2)-2*r1^3*(r2*Vr1+8*Vr3)-2*r1*(r2^3*...
    Vr1+r2*((-48+r3^2)*Vr1-16*r3*Vr2)+8*r2^2*Vr3-8*(16+r3^2)*Vr3));
JG12=16*(2*r2^3*(8*Vr1+r3*Vr2)-r2^4*Vr3-24*r2^2*(r1*Vr2-4*Vr3)+...
    (16+r1^2+r3^2)*(8*r1*Vr2+r1^2*Vr3+(-16+r3^2)*Vr3)-2*r2*(8*(16+...
    r1^2-r3^2)*Vr1-r3*((-48+r1^2+r3^2)*Vr2-16*r1*Vr3)));
JG22=32*(r1^3*(4*Vr1+r2*Vr3)+r1^2*(-r2*r3*Vr1+16*r2*Vr2+4*r3*Vr3)+...
    r3*(-r2^3*Vr1+r2*((48-r3^2)*Vr1+16*r3*Vr2)-12*r2^2*Vr3+4*(16+...
    r3^2)*Vr3)+r1*(4*(16-3*r2^2+r3^2)*Vr1+r2*(-48+r2^2+r3^2)*Vr3));
JG32=-16*((-256+r1^4+96*r2^2-r2^4+32*r1*r2*r3+2*r1^2*r3^2+r3^4)*...
    Vr1+2*(r1^3*r2*Vr2+r1*r2*(-48+r2^2+r3^2)*Vr2-4*r1^2*(r3*Vr2+2*...
    r2*Vr3)-4*((16-3*r2^2)*r3*Vr2+r3^3*Vr2+2*r2*(-16+r2^2)*Vr3-2*...
    r2*r3^2*Vr3)));
JG13=-16*(-r3^4*Vr2+24*r3^2*(4*Vr2+r1*Vr3)+(16+r1^2+r2^2)*((-16+...
    r1^2+r2^2)*Vr2-8*r1*Vr3)+2*r3^3*(-8*Vr1+r2*Vr3)+2*r3*(8*(16+...
    r1^2-r2^2)*Vr1+r2*(16*r1*Vr2+r1^2*Vr3+(-48+r2^2)*Vr3)));
JG23=16*((-256+r1^4+2*r1^2*r2^2+r2^4-32*r1*r2*r3+96*r3^2-r3^4)*...
    Vr1+2*(4*r2*(16+r1^2+r2^2)*Vr3-12*r2*r3^2*Vr3+r3^3*(8*Vr2+r1*...
```

```
    Vr3)+r3*(8*(-16+r1^2-r2^2)*Vr2+r1*(-48+r1^2+r2^2)*Vr3)));
JG33=-32*(r1^3*(-4*Vr1+r3*Vr2)+r1*(-4*(16+r2^2-3*r3^2)*Vr1+r3*...
    (-48+r2^2+r3^2)*Vr2)+r1^2*(r2*r3*Vr1+4*r2*Vr2+16*r3*Vr3)-r2*...
    (r3^3*Vr1+4*(16+r2^2)*Vr2-12*r3^2*Vr2+r3*(-48*Vr1+r2^2*Vr1...
    +16*r2*Vr3)));
JG=1/JGden*[JG11 JG12 JG13;
            JG21 JG22 JG23;
            JG31 JG32 JG33];

%Partial [C w]/ Partial r - damping terms
JHden=(16+r1^2+r2^2+r3^2)^3;
JH11=-32*(r1^3*(-r3*w2+r2*w3)+12*r1^2*(r2*w2+r3*w3)-4*(16+r2^2+...
    r3^2)*(r2*w2+r3*w3)+r1*(-r2^2*(16*w1+r3*w2)-r3*(16*r3*w1-48*w2...
    +r3^2*w2)+r2^3*w3+r2*(-48+r3^2)*w3));
JH21=16*(8*r2^3*w1-2*r1^3*(r3*w1-8*w2)-2*r1*(-48*r3*w1+r3^3*w1+...
    128*w2-8*r3^2*w2)-96*r1^2*w3+r1^4*w3-r2^4*w3-(-256+r3^4)*w3-8*...
    r2*((-16+3*r1^2-r3^2)*w1+4*r1*r3*w3)-2*r2^2*(r1*r3*w1+8*r1*w2+...
    r3^2*w3));
JH31=-16*(r1^4*w2+24*r1^2*(r3*w1-4*w2)-(16+r2^2+r3^2)*(8*r3*w1+(-16+...
    r2^2)*w2+r3^2*w2)-2*r1^3*(r2*w1+8*w3)-2*r1*(r2^3*w1+r2*((-48+...
    r3^2)*w1-16*r3*w2)+8*r2^2*w3-8*(16+r3^2)*w3));
JH12=16*(2*r2^3*(8*w1+r3*w2)-r2^4*w3-24*r2^2*(r1*w2-4*w3)+(16+r1^2+...
    r3^2)*(8*r1*w2+r1^2*w3+(-16+r3^2)*w3)-2*r2*(8*(16+r1^2-r3^2)*...
    w1-r3*((-48+r1^2+r3^2)*w2-16*r1*w3)));
JH22=32*(r1^3*(4*w1+r2*w3)+r1^2*(-r2*r3*w1+16*r2*w2+4*r3*w3)+r3*...
    (-r2^3*w1+r2*((48-r3^2)*w1+16*r3*w2)-12*r2^2*w3+4*(16+r3^2)*...
    w3)+r1*(4*(16-3*r2^2+r3^2)*w1+r2*(-48+r2^2+r3^2)*w3));
JH32=-16*((-256+r1^4+96*r2^2-r2^4+32*r1*r2*r3+2*r1^2*r3^2+r3^4)*w1+...
    2*(r1^3*r2*w2+r1*r2*(-48+r2^2+r3^2)*w2-4*r1^2*(r3*w2+2*r2*w3)-...
    4*((16-3*r2^2)*r3*w2+r3^3*w2+2*r2*(-16+r2^2)*w3-2*r2*r3^2*w3)));
JH13=-16*(-r3^4*w2+24*r3^2*(4*w2+r1*w3)+(16+r1^2+r2^2)*((-16+r1^2+...
    r2^2)*w2-8*r1*w3)+2*r3^3*(-8*w1+r2*w3)+2*r3*(8*(16+r1^2-r2^2)*...
    w1+r2*(16*r1*w2+r1^2*w3+(-48+r2^2)*w3)));
JH23=16*((-256+r1^4+2*r1^2*r2^2+r2^4-32*r1*r2*r3+96*r3^2-r3^4)*w1+...
    2*(4*r2*(16+r1^2+r2^2)*w3-12*r2*r3^2*w3+r3^3*(8*w2+r1*w3)+r3*...
    (8*(-16+r1^2-r2^2)*w2+r1*(-48+r1^2+r2^2)*w3)));
JH33=-32*(r1^3*(-4*w1+r3*w2)+r1*(-4*(16+r2^2-3*r3^2)*w1+r3*(-48+...
    r2^2+r3^2)*w2)+r1^2*(r2*r3*w1+4*r2*w2+16*r3*w3)-r2*(r3^3*w1+4*...
    (16+r2^2)*w2-12*r3^2*w2+r3*(-48*w1+r2^2*w1+16*r2*w3)));
JH=1/JHden*[JH11 JH12 JH13;
            JH21 JH22 JH23;
            JH31 JH32 JH33];

%%ORIGINAL EQUATIONS
%Fill the n x n block with non zero elements (diagonal block)
Jacobian((42*n-35):(42*n-33),(42*n-41):(42*n-39))=...
```

```
    dt/2*tilde(w)+eye(3);%delP1/q
Jacobian((42*n-32):(42*n-30),(42*n-41):(42*n-39))=...
    dt/2*tilde(w)-eye(3);%delP2/q
Jacobian((42*n-23):(42*n-21),(42*n-41):(42*n-39))=...
    -dx/2*tilde(k)-eye(3);%delF1/q
Jacobian((42*n-20):(42*n-18),(42*n-41):(42*n-39))=...
    -dx/2*tilde(k)+eye(3);%delF2/q
Jacobian((42*n-41):(42*n-39),(42*n-41):(42*n-39))=...
    dx*dt/4*SIMS.edf*C*tilde(w);%delq/q-with damping factor
Jacobian((42*n-35):(42*n-33),(42*n-38):(42*n-36))=...
    -dt/2*JA;%delP1/r
Jacobian((42*n-32):(42*n-30),(42*n-38):(42*n-36))=...
    -dt/2*JA;%delP2/r
Jacobian((42*n-29):(42*n-27),(42*n-38):(42*n-36))=...
    dt/2*(JB-JC)+eye(3);%delH1/r
Jacobian((42*n-26):(42*n-24),(42*n-38):(42*n-36))=...
    dt/2*(JB-JC)-eye(3);%delH2/r
Jacobian((42*n-23):(42*n-21),(42*n-38):(42*n-36))=...
    dx/2*JD;%delF1/r
Jacobian((42*n-20):(42*n-18),(42*n-38):(42*n-36))=...
    dx/2*JD;%delF2/r
Jacobian((42*n-17):(42*n-15),(42*n-38):(42*n-36))=...
    dx/2*(JF-JE)-eye(3);%delM1/r
Jacobian((42*n-14):(42*n-12),(42*n-38):(42*n-36))=...
    dx/2*(JF-JE)+eye(3);%delM2/r
Jacobian((42*n-41):(42*n-39),(42*n-38):(42*n-36))=...
    dx*dt/4*SIMS.edf*JG;%delq/r-with damping factor
Jacobian((42*n-38):(42*n-36),(42*n-38):(42*n-36))=...
    dx*dt/4*SIMS.edm*JH;%delpsi/r-with damping factor
Jacobian((42*n-32):(42*n-30),(42*n-35):(42*n-33))=eye(3);%delP2/q3
Jacobian((42*n-26):(42*n-24),(42*n-32):(42*n-30))=eye(3);%delH2/r3
Jacobian((42*n-41):(42*n-39),(42*n-29):(42*n-27))=...
    -dx/4*eye(3)-dx*dt/8*tilde(W);%delq/P2-dynamic
Jacobian((42*n-38):(42*n-36),(42*n-29):(42*n-27))=...
    -dx*dt/8*tilde(V);%delpsi/P2
Jacobian((42*n-11):(42*n-9),(42*n-29):(42*n-27))=...
    1/2*eye(3);%delV/P2-dynamic
Jacobian((42*n-38):(42*n-36),(42*n-26):(42*n-24))=...
    -dx/4*eye(3)-dx*dt/8*tilde(W);%delpsi/H2-dynamic
Jacobian((42*n-8):(42*n-6),(42*n-26):(42*n-24))=...
    1/2*eye(3);%delW/H2-dynamic
Jacobian((42*n-20):(42*n-18),(42*n-23):(42*n-21))=-eye(3);%delF2/q2
Jacobian((42*n-14):(42*n-12),(42*n-20):(42*n-18))=-eye(3);%delM2/r2
Jacobian((42*n-41):(42*n-39),(42*n-17):(42*n-15))=...
    -dt/4*eye(3)+dx*dt/8*tilde(K);%delq/F1
Jacobian((42*n-38):(42*n-36),(42*n-17):(42*n-15))=...
```

```
    dx*dt/8*tilde(e1+G);%delpsi/F1
Jacobian((42*n-5):(42*n-3),(42*n-17):(42*n-15))=...
    -1/2*eye(3);%delG/F1
Jacobian((42*n-38):(42*n-36),(42*n-14):(42*n-12))=...
    -dt/4*eye(3)+dx*dt/8*tilde(K);%delpsi/M1
Jacobian((42*n-2):(42*n),(42*n-14):(42*n-12))=...
    -1/2*eye(3);%delK/M1
Jacobian((42*n-41):(42*n-39),(42*n-11):(42*n-9))=...
    -SIMS.edf*dx*dt/4; %delq/delV-with damping factor
Jacobian((42*n-38):(42*n-36),(42*n-11):(42*n-9))=...
    dx*dt/8*tilde(P1+P2);%delpsi/V
Jacobian((42*n-35):(42*n-33),(42*n-11):(42*n-9))=-dt/2*C';%delP1/V
Jacobian((42*n-32):(42*n-30),(42*n-11):(42*n-9))=-dt/2*C';%delP2/V
Jacobian((42*n-11):(42*n-9),(42*n-11):(42*n-9))=-mass*eye(3);%delV/V
Jacobian((42*n-8):(42*n-6),(42*n-11):(42*n-9))=...
    -mass*tilde(r_cg);%delW/V
Jacobian((42*n-41):(42*n-39),(42*n-8):(42*n-6))=...
    dx*dt/8*tilde(P1+P2);%delq/W
Jacobian((42*n-38):(42*n-36),(42*n-8):(42*n-6))=...
    dx*dt/8*tilde(H1+H2)-dx*dt/4*SIMS.edm;%delpsi/W-with damping factor
Jacobian((42*n-29):(42*n-27),(42*n-8):(42*n-6))=-dt/2*inv(HH);%delH1/W
Jacobian((42*n-26):(42*n-24),(42*n-8):(42*n-6))=-dt/2*inv(HH);%delH2/W
Jacobian((42*n-11):(42*n-9),(42*n-8):(42*n-6))=...
    mass*tilde(r_cg);%delV/W
Jacobian((42*n-8):(42*n-6),(42*n-8):(42*n-6))=-I;%delW/W
Jacobian((42*n-38):(42*n-36),(42*n-5):(42*n-3))=...
    -dx*dt/8*tilde(F1+F2);%delpsi/G
Jacobian((42*n-23):(42*n-21),(42*n-5):(42*n-3))=dx/2*C';%delF1/G
Jacobian((42*n-20):(42*n-18),(42*n-5):(42*n-3))=dx/2*C';%delF2/G
Jacobian((42*n-5):(42*n-3),(42*n-5):(42*n-3))=A;%delG/G
Jacobian((42*n-2):(42*n),(42*n-5):(42*n-3))=B';%delK/G
Jacobian((42*n-41):(42*n-39),(42*n-2):(42*n))=...
    -dx*dt/8*tilde(F1+F2);%delq/K
Jacobian((42*n-38):(42*n-36),(42*n-2):(42*n))=...
    -dx*dt/8*tilde(M1+M2);%delpsi/K
Jacobian((42*n-17):(42*n-15),(42*n-2):(42*n))=dx/2*inv(HH);%delM1/K
Jacobian((42*n-14):(42*n-12),(42*n-2):(42*n))=dx/2*inv(HH);%delM2/K
Jacobian((42*n-5):(42*n-3),(42*n-2):(42*n))=B;%delG/K
Jacobian((42*n-2):(42*n),(42*n-2):(42*n))=D;%delK/K

%Contributions to off diagonal blocks
if n > 1
    %The n x (n-1) block
    Jacobian((42*n-23):(42*n-21),(42*(n-1)-23):(42*(n-1)-21))=...
        eye(3);%delF1(n)/q2(n-1),ie q4
    Jacobian((42*n-17):(42*n-15),(42*(n-1)-20):(42*(n-1)-18))=...
```

```
            eye(3);%delM1(n)/r2(n-1),ie r4
    end

    if n < SIMS.num_x
        %The n x (n+1) block
        Jacobian((42*n-41):(42*n-39),(42*(n+1)-17):(42*(n+1)-15))=...
            dt/4*eye(3)+dx*dt/8*tilde(K);%delq(n)/F(n+1), ie F2
        Jacobian((42*n-38):(42*n-36),(42*(n+1)-17):(42*(n+1)-15))=...
            dx*dt/8*tilde(e1+G);%delpsi(n)/F(n+1)
        Jacobian((42*n-5):(42*n-3),(42*(n+1)-17):(42*(n+1)-15))=...
            -1/2*eye(3);%delG(n)/F(n+1)
        Jacobian((42*n-38):(42*n-36),(42*(n+1)-14):(42*(n+1)-12))=...
            dt/4*eye(3)+dx*dt/8*tilde(K);%delpsi(n)/M(n+1), ie M2
        Jacobian((42*n-2):(42*n),(42*(n+1)-14):(42*(n+1)-12))=...
            -1/2*eye(3);%delK(n)/M(n+1)
    end

end
%%-------------------------------------------------------------------------%%
```

D.10 Milenkovic.m

This file is used to evaluate the direction cosine matrix (C^{Bb}) and the rotation matrix (H^{Bb}) using Milenkovic parameters ($\rho_{B/b}$). It can be modified for postprocessing to evaluate Euler rotations as well.

```
%%-------------------------------------------------------------------------%%
function [H,C]=Milenkovic(r);
%Weiner-Milenkovic Parameters for Rotations
r_bar=2-1/8*r'*r;
H=((2*r_bar+1/2*r'*r)*eye(3)-2*tilde(r)+1/2*tilde(r)*tilde(r))/(4-r_bar)^2;
C=H*inv(H)';
% %For Euler angle recovery
% Theta=arccos(C(3,3));
% Psi=arcsin(C(3,1)/sin(Theta));
% Phi=arcsin(C(1,3)/sin(Theta));
%%-------------------------------------------------------------------------%%
```

D.11 Tilde.m

The tilde function is used to perform the cross product operator, ($\tilde{}$), by accepting a vector and returning its measure numbers as a skew symmetric matrix.

```
%%-------------------------------------------------------------------------%%
function T=tilde(X);
%Accepts 3x1 vector X and returns a skew symmetric 3x3 matrix T
T=[0 -X(3) X(2);X(3) 0 -X(1);-X(2) X(1) 0];
%%-------------------------------------------------------------------------%%
```

D.12 ImportStatic.m

This is a sample file used for postprocessing steady-state data. Ultimately, a different file was used for each individual problem, but most follow this basic guideline.

```
%%-------------------------------------------------------------------------%%
% Post processor for static problems
% clear;clc;
global SIMS

% Import from dat files for postprocessing
XStatic=dlmread('XStatic.dat','\t');
% XStatic=dlmread('72in20elemtrans1x1000XStatic.dat','\t');
SIMS.num_x=length(XStatic(:,1))/42; Taperrod24
SIMS.dx=GEOM.length/SIMS.num_x;
SIMS.num_t=1;   %temporarily
Simparams

% Steady Analysis Results
Centroid0(1)=SIMS.dx/2; Edges0(1)=0; for n=1:SIMS.num_x;
    % Properties in middle of nth elements (centroid)
    qbar(:,n)=XStatic((42*n-41):(42*n-39),1);
    rbar(:,n)=XStatic((42*n-38):(42*n-36),1);
    Vbar(:,n)=XStatic((42*n-11):(42*n-9),1);
    Wbar(:,n)=XStatic((42*n-8):(42*n-6),1);
    Gbar(:,n)=XStatic((42*n-5):(42*n-3),1);
    Kbar(:,n)=XStatic((42*n-2):(42*n),1)-GEOM.k_b(:,n);
    Pbar(:,n)=XStatic((42*n-29):(42*n-27),1);
    Hbar(:,n)=XStatic((42*n-26):(42*n-24),1);
    if n == SIMS.num_x,
        Fbar(:,n)=1/2*(BC.F2_B0(:,SIMS.num_x)+...
            XStatic((42*n-17):(42*n-15),1));
        Mbar(:,n)=1/2*(BC.M2_B0(:,SIMS.num_x)+...
            XStatic((42*n-14):(42*n-12),1));
    else
        Fbar(:,n)=1/2*(XStatic((42*(n+1)-17):(42*(n+1)-15),1)+...
            XStatic((42*n-17):(42*n-15),1));
        Mbar(:,n)=1/2*(XStatic((42*(n+1)-14):(42*(n+1)-12),1)+...
            XStatic((42*n-14):(42*n-12),1));
```

```
    end
    if n < SIMS.num_x;
        Centroid0(n+1)=Centroid0(n)+SIMS.dx;
    end
    % Properties from 1 to n+1 element edges
    qhat(:,n+1)=XStatic((42*n-23):(42*n-21),1);
    rhat(:,n+1)=XStatic((42*n-20):(42*n-18),1);
    Fhat(:,n)=XStatic((42*n-17):(42*n-15),1);
    Mhat(:,n)=XStatic((42*n-14):(42*n-12),1);
    Edges0(n+1)=Edges0(n)+SIMS.dx;
end
%Include Boundary Conditions
Fhat(:,SIMS.num_x+1)=BC.F2_B0(:,SIMS.num_x);
Mhat(:,SIMS.num_x+1)=BC.M2_B0(:,SIMS.num_x);
qhat(:,1)=BC.q4_b0(:,1); rhat(:,1)=BC.r4_b0(:,1);

% Display static results on screen
Fhat Mhat qhat rhat

figure(1) title('Static Resultants') subplot 211;
plot(Centroid0,Fbar');hold on plot(Edges0,Fhat');hold off
xlabel('X Location (inches)'); ylabel('Force (Mlb)');
title('Internal Forces (Positive Tension)');

subplot 212; plot(Centroid0,Mbar');hold on plot(Edges0,Mhat');hold
off xlabel('X Location (inches)'); ylabel('Moment (in-Mlb)');
title('Internal Moments');

figure(2) title('Static Deformation') subplot 211
plot(Centroid0,qbar');hold on plot(Edges0,qhat');hold off
xlabel('X Location (inches)'); ylabel('Deformation (inches)');
title('Deformation');

subplot 212 plot(Centroid0,rbar');hold on plot(Edges0,rhat');hold
off xlabel('X Location (inches)'); ylabel('Deformation (rad)');
title('Rotation');

figure(3) title('Static Strains'); subplot 211
plot(Centroid0,Gbar'); xlabel('X Location (inches)');
ylabel('Force Strain (-)'); title('Force Strains');

subplot 212 plot(Centroid0,Kbar'); xlabel('X Location (inches)');
ylabel('Moment Strain (-)'); title('Moment Strains');

figure(4) subplot 211 plot(Centroid0,Vbar'); xlabel('X Location
(inches)'); ylabel('Velocity (in/ms)'); title('Initial Velocity
```

```
Profile');

subplot 212 plot(Centroid0',Wbar); xlabel('X Location (inches)');
ylabel('Angular velocity (rad/ms)'); title('Initial Angular
Velocity Profile');
%%-------------------------------------------------------------------------%%
```

D.13 ImportDynamic.m

This is a sample file used to postprocess time accurate data. As with steady-state problems, individual scenarios require specialized files for postprocessing, but generally follow these guidelines.

```
%%-------------------------------------------------------------------------%%
% Post processor fordynamic problems~ use after ImportStatic
global SIMS

% Import from dat files for postprocessing
XDynamic=dlmread('dynamic.dat','\t');
% XDynamic=dlmread('72in20elemtrans1x1000dynamic.dat','\t');
SIMS.num_t=length(XDynamic(1,:));
SIMS.time=XDynamic(42*SIMS.num_x+2,SIMS.num_t);
SIMS.dt=SIMS.time/SIMS.num_t; Simparams

% Edge times
time(1)=0; time(2:(SIMS.num_t+1))=XDynamic(42*SIMS.num_x+2,:);


%Force Array
for t=1:SIMS.num_t
    for n=1:SIMS.num_x
        Fhat3(n,1)=XStatic(42*n-15,1);
        Fhat3(n,t+1)=XDynamic(42*n-15,t);
    end
    Fhat3(SIMS.num_x+1,t)=BC.F2_B(3,SIMS.num_x,t);
end

%Force Array
for t=1:SIMS.num_t
    for n=1:SIMS.num_x
        Fhat1(n,1)=XStatic(42*n-17,1);
        Fhat1(n,t+1)=XDynamic(42*n-17,t);
    end
    Fhat1(SIMS.num_x+1,t)=BC.F2_B(1,SIMS.num_x,t);
```

```
end


%Moment Array
for t=1:SIMS.num_t
    for n=1:SIMS.num_x
        Mhat1(n,1)=XStatic(42*n-14,1);
        Mhat1(n,t+1)=XDynamic(42*n-14,t);
    end
    Mhat1(SIMS.num_x+1,t)=BC.M2_B(1,SIMS.num_x,t);
end



% Tip Displacement
figure(5)
deflection(:,1)=XStatic((42*SIMS.num_x-23):(42*SIMS.num_x-21),1);
deflection(:,2:(SIMS.num_t+1))=...
    XDynamic((42*SIMS.num_x-23):(42*SIMS.num_x-21),:);
plot(time,deflection(1,:),'rx');title('Tip Deflection');

%Tip velocity
figure(6)
velocity(:,1)=XStatic((42*SIMS.num_x-11):(42*SIMS.num_x-9),1);
velocity(:,2:(SIMS.num_t+1))=...
    XDynamic((42*SIMS.num_x-11):(42*SIMS.num_x-9),:);
plot(time,velocity);legend('x','y','z');title('Tip Velocity')

%Tip slope
figure(7)
slope(:,1)=XStatic((42*SIMS.num_x-20):(42*SIMS.num_x-18),1);
slope(:,2:(SIMS.num_t+1))=...
    XDynamic((42*SIMS.num_x-20):(42*SIMS.num_x-18),:);
plot(time,slope);legend('p1','p2','p3');title('Tip Slope')

%Root Moment
figure(8); rootmoment(:,1)=XStatic(28:30,1);
rootmoment(:,2:(SIMS.num_t+1))=XDynamic(28:30,:);
plot(time,rootmoment);legend('x','y','z');title('Root moment')

%Tip momentum
figure(9)
momentum(:,1)=XStatic((42*SIMS.num_x-29):(42*SIMS.num_x-24),1);
momentum(:,2:(SIMS.num_t+1))=...
    XDynamic((42*SIMS.num_x-29):(42*SIMS.num_x-24),:);
plot(time,momentum); legend('x','y','z','rx','ry','rz');title('Tip
```

```
Momentum')

%Root Forces
figure(11) rforces(:,1)=XStatic((42*1-17):(42*1-15),1);
rforces(:,2:(SIMS.num_t+1))=XDynamic((42*1-17):(42*1-15),:);
plot(time,rforces);legend('x','y','z');title('Root forces')

% Frequency Response
sampletime=SIMS.dt; %ms
Fs=1000/sampletime; %Hz
[Pxx,F]=pwelch(deflection(1,:),[],[],8192,Fs); figure(12)
semilogx(F,Pxx); xlabel('Frequency (Hz)'); ylabel('Signal
Power(dB)'); title('Power Spectral Density');
[r,c]=find(Pxx==max(Pxx)); Fpeak=F(r,c);

figure(13) surf(time,Edges0,Fhat1);
xlabel('time');ylabel('space');zlabel('force');
%%-------------------------------------------------------------------------%%
```

Bibliography

1. Amirouche, Farid M. L. *Computational Methods in Multibody Dynamics*. Prentice Hall, New Jersey, 1992.

2. Argyris, J. H. and D. W. Scharpf. "Finite elements in time and space". *Aeronautical Journal of the Royal Society*, 73:1041–1044, 1969.

3. Atilgan, A. R. and D. H. Hodges. "Space-Time Mixed Finite Elements for Rods". *Journal of Sound and Vibration*, 192(3):731–739, 1996.

4. Atilgan, Ali R. and Dewey H. Hodges. "A Geometrically Nonlinear Analysis for Nonhomogeneous, Anisotropic Beams". *Proceedings of the 30th Structures, Structural Dynamics and Materials Conference, Part 2B*. American Institute of Aeronautics and Astronautics, Mobile, Alabama, 1989.

5. Bailey, Cecil D. "Application of Hamilton's Law of Varying Action". *AIAA Journal*, 13(9):1154–1157, September 1975.

6. Bailey, Cecil D. "A New Look at Hamilton's Principle". *Foundations of Physics*, 5(3):433–451, March 1975.

7. Bailey, Cecil D. "The Method of Ritz Applied to the Equation of Hamilton". *Computer Methods in Applied Mechanics and Engineering*, 7(2):235–247, 1976.

8. Bailey, Cecil D. "Direct Analytical Solutions to Non-Uniform Beam Problems". *Journal of Sound and Vibration*, 56(4):501–507, 1978.

9. Baruch, Menahem and Richard Riff. "Hamilton's Principle, Hamilton's Law-6n Correct Formulations". *AIAA Journal*, 20(5):687–692, May 1982.

10. Bauchau, O. A. and N. J. Theron. "Energy Decaying Schemes for Nonlinear Beam Models". *Computer Methods in Applied Mechanics and Engineering*, 134:37–56, 1996.

11. Bauchau, Olivier A., Carlo L. Bottasso, and Lorenzo Trainelli. "Robust integration schemes for flexible multibody systems". *Computer Methods in Applied Mechanics and Engineering*, 192(3):395–420, 2003.

12. Bauchau, Olivier A. and Lorenzo Trainelli. "The Vectorial Parameterization of Rotation". *Nonlinear Dynamics*, 32:71–92, 2003.

13. Belytschko, T. and B.J. Hsieh. "Nonlinear transient finite element analysis with convected coordinates". *International Journal of Numerical Methods in Engineering*, 7:255–271, 1973.

14. Borri, M., G. L. Ghiringhelli, M. Lanz, P. Mantegazza, and T. Merlini. "Dynamic Response of Mechanical Systems by A Weak Hamiltonian Formulation". *Computers and Structures*, 20:495–508, 1985.

15. Borri, M., F. Mello, M. Iura, and S. N. Atluri. "Primal and Mixed Forms of Hamilton's Principle for Constrained and Flexible Dynamical Systems: Numerical Studies". *ARO/AFOSR Workshop on Nonlinear Dynamics*. Virginia Polytechnic Institute and State University, Blacksburg, VA, June 1-3 1988.

16. Canavin, J. R. and P. W. Likins. "Floating Reference Frames for Flexible Spacecraft". *Journal of Spacecraft and Rockets*, 14(12):724–732, 1977.

17. Cook, Robert D., David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and Applications of Finite Element Analysis, Fourth Edition*. John Wiley and Sons, Inc., United States, 1974.

18. Danielson, D. A. and D. H. Hodges. "Nonlinear Beam Kinematics by Decomposition of the Rotation Tensor". *Journal of Applied Mechanics*, 54:258–262, June 1987.

19. Danielson, D. A. and D. H. Hodges. "A Beam Theory for Large Global Rotation, Moderate Local Rotation, and Small Strain". *Journal of Applied Mechanics*, 55:179–184, March 1988.

20. Downer, J.D., K.C. Park, and J.C. Chou. "Dynamics of flexible beams for multibody systems: A computational procedure". *Computer Methods in Applied Mechanics and Engineering*, 96:373–408, 1992.

21. Fowles, Grant R. and George L. Cassiday. *Analytical Mechanics (Sixth Edition)*. Harcourt, Inc., Orlando, 1999.

22. Fried, I. "Finite element analysis of time dependent phenomena". *AIAA Journal*, 7:1170–1173, 1969.

23. Grohmann, Boris A., Thomas Wallmersperger, and Bernd-Helmut Kroplin. "Time-Discontinuous Stabilized Space-Time Finite Elements for Timoshenko Beams". *AIAA Journal*, 39(11):2158–2166, November 2001.

24. Haug, E. J. *Computer Aided Kinematics and Dynamics of Mechanical Systems, Volume 1: Basic Methods*. Allyn and Bacon, 1989.

25. Hibbeler, R. C. *Mechanics of Materials, Fourth Edition*. Prentice Hall, New Jersey, 2000.

26. Hodges, D. H. "A Mixed Variational Formulation Based on Exact Intrinsic Equations for Dynamics of Moving Beams". *International Journal of Solids and Structures*, 14(11):1253–1273, November 1990.

27. Hodges, Dewey H. "Some Fundamentals Regarding Kinematics and Generalized Forces for Multibody Dynamics". *Journal of the American Helicopter Society*, 3–11, July 1990.

28. Hodges, Dewey H. "Geometrically Exact, Intrinsic Theory for Dynamics of Curved and Twisted Anisotropic Beams". *AIAA Journal*, 41(6):1131–1137, June 2003.

29. Hodges, Dewey H. and Robert R. Bless. "Weak Hamiltonian Finite Element Method for Optimal Control Problems". *Journal of Guidance, Control, and Dynamics*, 14(1):148–155, 1991.

30. Hopkins, A. Stewart and Robert A. Ormiston. "An Examination of Selected Problems in Rotor Blade Structural Mechanics and Dynamics". *The 59th Annual Forum of the American Helicopter Society*. The American Helicopter Society International, Phoenix, AZ, May 2003.

31. Hou, L. J. and D. A. Peters. "Application of Triangular Space-Time Finite Elements to Problems of Wave Propagation". *Journal of Sound and Vibration*, 173(5):611–632, 1994.

32. Housner, J. "Convected transient analysis for large space structure maneuver and deployment". *Proceedings of the 25th Structures, Structural Dynamics and Materials Conference*, 616–629. American Institute of Aeronautics and Astronautics, 1984.

33. Housner, J. M., S. C. Wu, and C. W. Chang. "A finite element method for time varying geometry in multibody structures". *Proceedings of the 29th Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics, 1988. AIAA Paper No. 88-2234.

34. Hughes, Thomas J. R. and Gregory M. Hulbert. "Space-Time Finite Element Methods for Elastodynamics: Formulations and Error Estimates". *Computer Methods in Applied Mechanics and Engineering*, 66:339–363, 1988.

35. J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, Philadelphia, 1996.

36. Kane, T. R., R. R. Ryan, and A. K. Banerjee. "Dynamics of a Cantilever Beam Attached to a Moving Base". *Journal of Guidance, Control, and Dynamics*, 10(2):139–1151, March 1987.

37. Kane, Thomas R. and David A. Levinson. *Dynamics: Theory and Applications*. McGraw-Hill Book Company, New York, NY, 1985.

38. Kunz, Donald L. "Multibody System Analysis Based on Hamilton's Weak Principle". *AIAA Journal*, 39(12):2382–2388, December 2001.

39. Kunz, Donald L. "Implementation of a Generalized Multibody Approach for Analysis of Dynamic Systems". *Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics, Austin, TX, April 2005. AIAA Paper No. 2005-2348.

40. Meirovitch, Leonard. *Methods of Analytical Dynamics*. McGraw Hill, New York, 1970.

41. Oden, J. T. "A generalized theory of finite elements. Part I: topological considerations. Part II: applications". *International Journal of Numerical Methods in Engineering*, 1:205–221,247–259, 1969.

42. Park, K. C. and J. C. Chiou. "Evaluation of constraint stabilization procedures for multibody dynamical systems". *Proceedings of the 28th Structures, Structural Dynamics and Materials Conference, Part 2B*. American Institute of Aeronautics and Astronautics, Monterey, CA, 1987. AIAA Paper No. 87-0927.

43. Park, K. C. and J. C. Chiou. "Stabilization of computational procedures for constrained dynamical systems". *Journal of Guidance, Control, and Dynamics*, 11:365–370, 1988.

44. Park, K. C., J. C. Chiou, and J. D. Downer. "Explicit-Implicit staggered procedure for multibody dynamics analysis". *Journal of Guidance, Control, and Dynamics*, 13:562–570, 1990.

45. Peters, D. A. and A. P. Izadpanah. "hp-version finite elements for the space time domain". *Computational Mechanics*, (3):73–88, 1988.

46. Riff, Richard and Menahem Baruch. "Stability of Time Finite Elements". *AIAA Journal*, 22(8):1171–1173, August 1984.

47. Riff, Richard and Menahem Baruch. "Time Finite Element Discretization of Hamilton's Law of Varying Action". *AIAA Journal*, 22(9):1310–1317, September 1984.

48. Roberson, R. E. and R. Schwertassek. *Dynamics of Multibody Systems*. Springer-Verlag, Germany, 1988.

49. Ryan, R. R. *Multibody Systems Handbook*. Springer-Verlag, Berlin, 1990.

50. Shabana, Ahmed A. *Dynamics of Multibody Systems*. Cambridge University Press, United Kingdom, 1998. Second Edition.

51. Simkins, T. E. "Finite Elements for Initial Value Problems in Dynamics". *AIAA Journal*, 10(10):1357–1362, October 1981.

52. Simo, J. C. "A finite strain beam formulation. The three dimensional dynamic problem. Part I". *Computer Methods in Applied Mechanics and Engineering*, 49:55–70, 1985.

53. Simo, J. C. and L. Vu-Quoc. "A three dimensional finite strain rod model. Part II: Computational Aspects". *Computer Methods in Applied Mechanics and Engineering*, 58:79–116, 1986.

54. Simo, J. C. and L. Vu-Quoc. "On the dynamics in space of rods undergoing large motions - A geometrically exact approach". *Computer Methods in Applied Mechanics and Engineering*, 66:125–161, 1988.

55. Smith, Gerald M. and Glenn L. Downey. *Advanced Engineering Dynamics*. International Textbook Company, Pennsylvania, 1968.

56. Wells, Dare A. *Theory and Problems of Lagrangian Dynamics.* McGraw-Hill, New York, 1967.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
23 Mar 2006	**Master's Thesis**	1 Sep 04 – 23 Mar 06

4. TITLE AND SUBTITLE

Simulation of a Moving, Elastic Beam using Hamilton's Weak Principle

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Leigh, Elliott J., 1st Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GAE/ENY/06-M21

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/VASD
Attn: Dr. Phil Beran
2210 8th St
WPAFB OH 45433-7765 DSN: 785-6645

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Hamilton's Law is derived in weak form for slender beams with closed cross sections. The result is discretized with mixed space-time finite elements to yield a system of nonlinear, algebraic equations. An algorithm is proposed for solving these equations using unconstrained optimization techniques, obtaining steady-state and time accurate solutions for problems of structural dynamics. This technique provides accurate solutions for nonlinear static and steady-state problems including the cantilevered elastica and flatwise rotation of beams. Modal analysis of beams and rods is investigated to accurately determine fundamental frequencies of vibration, and the simulation of simple maneuvers is demonstrated.

15. SUBJECT TERMS

Finite element analysis, dynamic response, beam equations, nonlinear algebraic equations, multibody dynamics, Hamilton's Weak Principle, Hamilton's Law, structural dynamics

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Dr. Donald L. Kunz
U	**U**	**U**	**UU**	190	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 785-3636, ext 4548 e-mail: Donald.kunz@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18